DonorsChoose

DonorsChoose.org receives hundreds of thousands of project proposals each year for classroom projects in need of funding. Right now, a large number of volunteers is needed to manually screen each submission before it's approved to be posted on the DonorsChoose.org website.

Next year, DonorsChoose.org expects to receive close to 500,000 project proposals. As a result, there are three main problems they need to solve:

- How to scale current manual processes and resources to screen 500,000 projects so that they can be posted as quickly and as efficiently as possible
- How to increase the consistency of project vetting across different volunteers to improve the experience for teachers
- How to focus volunteer time on the applications that need the most assistance

The goal of the competition is to predict whether or not a DonorsChoose.org project proposal submitted by a teacher will be approved, using the text of project descriptions as well as additional metadata about the project, teacher, and school. DonorsChoose.org can then use this information to identify projects most likely to need further review before approval.

About the DonorsChoose Data Set

The `train.csv` data set provided by DonorsChoose contains the following features:

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example:** `p036502`
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy!` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targeted. One of the following enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories for the project from the following enumerated list of values: • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`
`school_state`	State where school is located (Two-letter U.S. postal code). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** • `Literacy`

Feature	Description
	• Literature & Writing, Social Sciences
project_resource_summary	An explanation of the resources needed for the project. **Example:** • My students need hands on literacy materials to manage sensory needs!
project_essay_1	First application essay*
project_essay_2	Second application essay*
project_essay_3	Third application essay*
project_essay_4	Fourth application essay*
project_submitted_datetime	Datetime when project application was submitted. **Example:** 2016-04-28 12:43:56.245
teacher_id	A unique identifier for the teacher of the proposed project. **Example:** bdf8baa8fedef6bfeec7ae4ff1c15c56
teacher_prefix	Teacher's title. One of the following enumerated values: • nan • Dr. • Mr. • Mrs. • Ms. • Teacher.
teacher_number_of_previously_posted_projects	Number of project applications previously submitted by the same teacher. **Example:** 2

* See the section **Notes on the Essay Data** for more details about these features.

Additionally, the resources.csv data set provides more data about the resources required for each project. Each line in this file represents a resource required by a project:

Feature	Description
id	A project_id value from the train.csv file. **Example:** p036502
description	Desciption of the resource. **Example:** Tenor Saxophone Reeds, Box of 25
quantity	Quantity of the resource required. **Example:** 3
price	Price of the resource required. **Example:** 9.95

Note: Many projects require multiple resources. The id value corresponds to a project_id in train.csv, so you use it as a key to retrieve all resources needed for a project:

The data set contains the following label (the value you will attempt to predict):

Label	Description
project_is_approved	A binary flag indicating whether DonorsChoose approved the project. A value of 0 indicates the project was not approved, and a value of 1 indicates the project was approved.

Notes on the Essay Data

Prior to May 17, 2016, the prompts for the essays were as follows:

- __project_essay_1:__ "Introduce us to your classroom"
- __project_essay_2:__ "Tell us more about your students"
- __project_essay_3:__ "Describe how your students will use the materials you're requesting"
- __project_essay_4:__ "Close by sharing why your project will make a difference"

Starting on May 17, 2016, the number of essays was reduced from 4 to 2, and the prompts for the first 2 essays were changed to the following:

- __project_essay_1:__ "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."

your neighborhood, and your school are all helpful.

- __project_essay_2:__ "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

  For all projects with project_submitted_datetime of 2016-05-17 and later, the values of project_essay_3 and project_essay_4 will be NaN.

In [3]:

```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import roc_curve, auc
from nltk.stem.porter import PorterStemmer

import re
# Tutorial about Python regular expressions: https://pymotw.com/2/re/
import string
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer

from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle

from tqdm import tqdm
import os

from plotly import plotly
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
from collections import Counter
```

```
C:\Users\samar\Anaconda3\lib\site-packages\gensim\utils.py:1197: UserWarning: detected Windows; al
iasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

1.1 Reading Data

In [4]:

```
#project_data = pd.read_csv('train_data.csv')
project_data =
pd.read_csv('D:\\Studies\\Hadoop\\Python\\AppliedAICourse\\Notes\\17_REAL_PROBLEM_PREDICT_RATING_AM
ZON\\ASSIGNMENT-2 Apply t-SNE\\Assignments_DonorsChoose\\train_data.csv')
#resource_data = pd.read_csv('resources.csv')
resource_data =
pd.read_csv('D:\\Studies\\Hadoop\\Python\\AppliedAICourse\\Notes\\17_REAL_PROBLEM_PREDICT_RATING_AM
ZON\\ASSIGNMENT-2 Apply t-SNE\\Assignments_DonorsChoose\\resources.csv')
```

In [20]:

```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
project_data.head(3)
```

```
Number of data points in train data (109248, 17)
---------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

Out[20]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Gra
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Gra
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	2016-08-31 12:03:56	Gra

print("Number of data points in train data", resource_data.shape) print(resource_data.columns.values) resource_data.head(2)

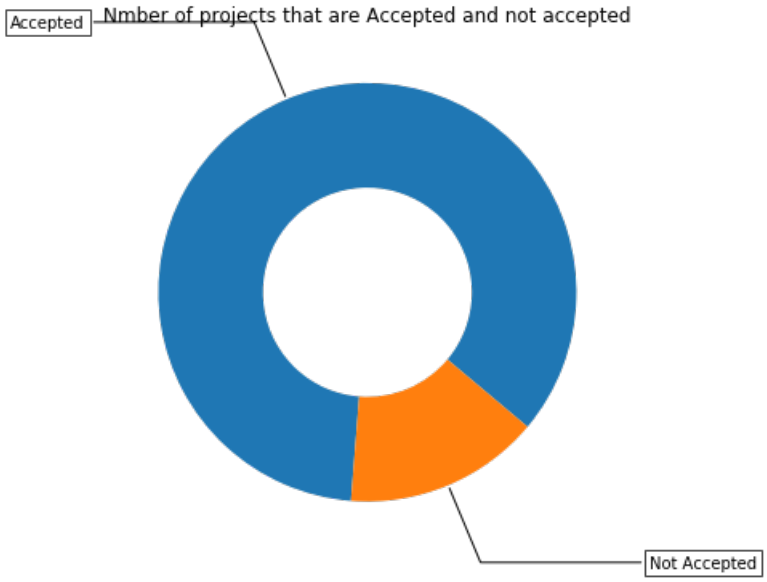
1.2 Data Analysis

In [21]:

```
# PROVIDE CITATIONS TO YOUR CODE IF YOU TAKE IT FROM ANOTHER WEBSITE.
# https://matplotlib.org/gallery/pie_and_polar_charts/pie_and_donut_labels.html#sphx-glr-gallery-pie-and-polar-charts-pie-and-donut-labels-py


y_value_counts = project_data['project_is_approved'].value_counts()
print("Number of projects thar are approved for funding ", y_value_counts[1], ", (",
(y_value_counts[1]/(y_value_counts[1]+y_value_counts[0]))*100,"%)")
print("Number of projects thar are not approved for funding ", y_value_counts[0], ", (",
(y_value_counts[0]/(y_value_counts[1]+y_value_counts[0]))*100,"%)")

fig, ax = plt.subplots(figsize=(6, 6), subplot_kw=dict(aspect="equal"))
recipe = ["Accepted", "Not Accepted"]

data = [y_value_counts[1], y_value_counts[0]]

wedges, texts = ax.pie(data, wedgeprops=dict(width=0.5), startangle=-40)

bbox_props = dict(boxstyle="square,pad=0.3", fc="w", ec="k", lw=0.72)
kw = dict(xycoords='data', textcoords='data', arrowprops=dict(arrowstyle="-"),
          bbox=bbox_props, zorder=0, va="center")

for i, p in enumerate(wedges):
    ang = (p.theta2 - p.theta1)/2. + p.theta1
    y = np.sin(np.deg2rad(ang))
    x = np.cos(np.deg2rad(ang))
    horizontalalignment = {-1: "right", 1: "left"}[int(np.sign(x))]
    connectionstyle = "angle,angleA=0,angleB={}".format(ang)
    kw["arrowprops"].update({"connectionstyle": connectionstyle})
    ax.annotate(recipe[i], xy=(x, y), xytext=(1.35*np.sign(x), 1.4*y),
                horizontalalignment=horizontalalignment, **kw)
```

```
                    horizontalalignment=horizontalalignment, **kw)

ax.set_title("Nmber of projects that are Accepted and not accepted")

plt.show()
```

```
Number of projects thar are approved for funding  92706 , ( 84.85830404217927 %)
Number of projects thar are not approved for funding  16542 , ( 15.141695957820739 %)
```



1.2.1 Univariate Analysis: School State

In [22]:

```
# Pandas dataframe groupby count, mean: https://stackoverflow.com/a/19385591/4084039

temp = pd.DataFrame(project_data.groupby("school_state")
["project_is_approved"].apply(np.mean)).reset_index()
# if you have data which contain only 0 and 1, then the mean = percentage (think about it)
temp.columns = ['state_code', 'num_proposals']

# How to plot US state heatmap: https://datascience.stackexchange.com/a/9620

scl = [[0.0, 'rgb(242,240,247)'],[0.2, 'rgb(218,218,235)'],[0.4, 'rgb(188,189,220)'],\
            [0.6, 'rgb(158,154,200)'],[0.8, 'rgb(117,107,177)'],[1.0, 'rgb(84,39,143)']]

data = [ dict(
        type='choropleth',
        colorscale = scl,
        autocolorscale = False,
        locations = temp['state_code'],
        z = temp['num_proposals'].astype(float),
        locationmode = 'USA-states',
        text = temp['state_code'],
        marker = dict(line = dict (color = 'rgb(255,255,255)',width = 2)),
        colorbar = dict(title = "% of pro")
    ) ]

layout = dict(
        title = 'Project Proposals % of Acceptance Rate by US States',
        geo = dict(
            scope='usa',
            projection=dict( type='albers usa' ),
            showlakes = True,
            lakecolor = 'rgb(255, 255, 255)',
        ),
    )

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='us-map-heat-map')
```

In [23]:

```
# https://www.csi.cuny.edu/sites/default/files/pdf/administration/ops/2letterstabbrev.pdf
temp.sort_values(by=['num_proposals'], inplace=True)
print("States with lowest % approvals")
print(temp.head(5))
print('='*50)
print("States with highest % approvals")
print(temp.tail(5))
```

```
States with lowest % approvals
   state_code  num_proposals
46         VT       0.800000
7          DC       0.802326
43         TX       0.813142
26         MT       0.816327
18         LA       0.831245
==================================================
States with highest % approvals
   state_code  num_proposals
30         NH       0.873563
35         OH       0.875152
47         WA       0.876178
28         ND       0.888112
8          DE       0.897959
```

In [24]:

```
#stacked bar plots matplotlib:
https://matplotlib.org/gallery/lines_bars_and_markers/bar_stacked.html
def stack_plot(data, xtick, col2='project_is_approved', col3='total'):
    ind = np.arange(data.shape[0])

    #print("="*50)
    #print(data.shape[0])
    #print("="*50)
    # https://medium.com/python-pandemonium/data-visualization-in-python-bar-graph-in-matplotlib-f
1738602e9c4
    # arange is numpymethod that generates an array of sequential numbers. For instance
numpy.arange(5)
    # will generate a numpy.ndArray like [0,1,2,3,4,5]
    # Why is it needed? We need some data for X-axis and right now we only have labels that can't
be used
    # for plotting purpose. So we will generate an array of length of label and use it on X-axis
```

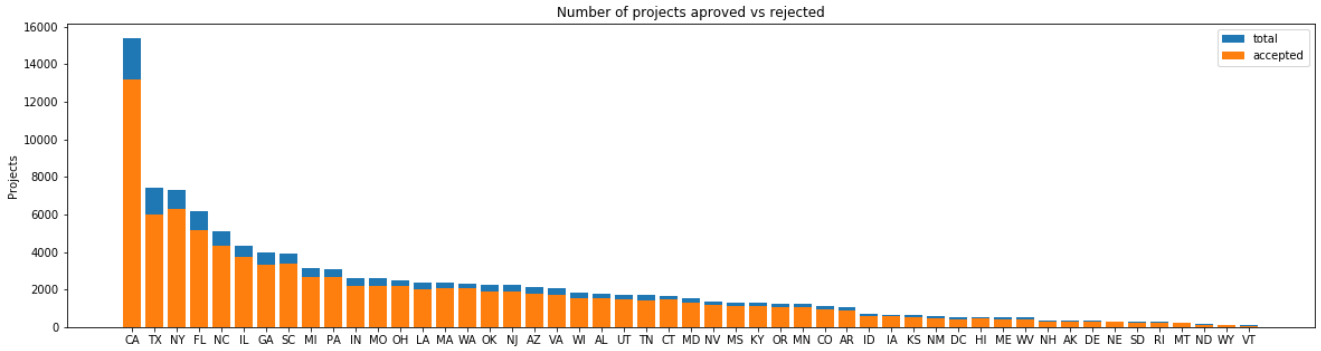
```
    plt.figure(figsize=(20,5))
    p1 = plt.bar(ind, data[col3].values) # total
    p2 = plt.bar(ind, data[col2].values) # project is accepted

    plt.ylabel('Projects')
    plt.title('Number of projects aproved vs rejected')
    plt.xticks(ind, list(data[xtick].values))
    plt.legend((p1[0], p2[0]), ('total', 'accepted'))
    plt.show()
```

In [25]:

```
def univariate_barplots(data, col1, col2='project_is_approved', top=False):
    # Count number of zeros in dataframe python: https://stackoverflow.com/a/51540521/4084039
    temp = pd.DataFrame(project_data.groupby(col1)[col2].agg(lambda x: x.eq(1).sum())).reset_index(
)

    # Pandas dataframe grouby count: https://stackoverflow.com/a/19385591/4084039
    temp['total'] = pd.DataFrame(project_data.groupby(col1)
[col2].agg({'total':'count'})).reset_index()['total']
    temp['Avg'] = pd.DataFrame(project_data.groupby(col1)[col2].agg({'Avg':'mean'})).reset_index()[
'Avg']

    temp.sort_values(by=['total'],inplace=True, ascending=False)
    #print("="*50)
    #print(temp.head(2))
    #print("="*50)
    if top:
        temp = temp[0:top]

    stack_plot(temp, xtick=col1, col2=col2, col3='total')
    print(temp.head(5))
    print("="*50)
    print(temp.tail(5))
```

In [26]:

```
univariate_barplots(project_data, 'school_state', 'project_is_approved', False)
```



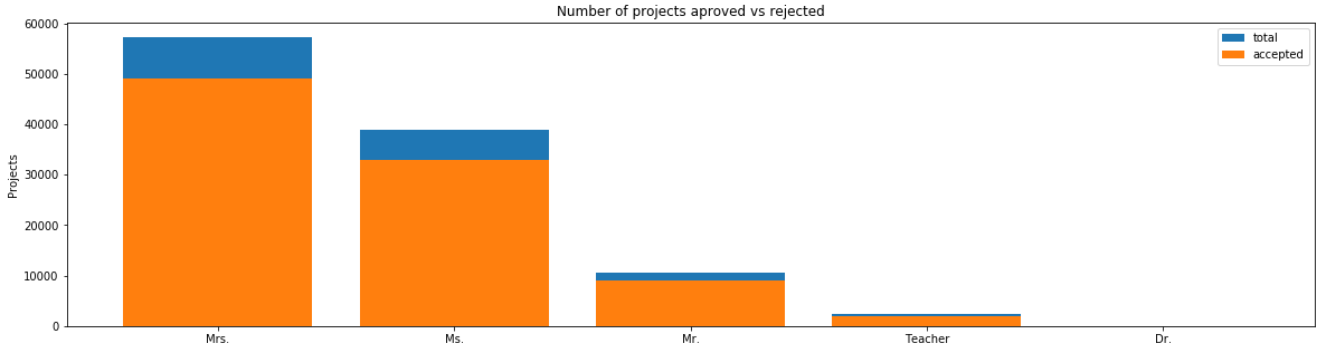

```
   school_state  project_is_approved  total       Avg
4            CA                13205  15388  0.858136
43           TX                 6014   7396  0.813142
34           NY                 6291   7318  0.859661
9            FL                 5144   6185  0.831690
27           NC                 4353   5091  0.855038
==================================================
   school_state  project_is_approved  total       Avg
39           RI                  243    285  0.852632
26           MT                  200    245  0.816327
28           ND                  127    143  0.888112
50           WY                   82     98  0.836735
46           VT                   64     80  0.800000
```

SUMMARY: Every state has greater than 80% success rate in approval

1.2.2 Univariate Analysis: teacher_prefix

In [27]:

```
univariate_barplots(project_data, 'teacher_prefix', 'project_is_approved' , top=False)
```




```
  teacher_prefix  project_is_approved  total       Avg
2           Mrs.                48997  57269  0.855559
3            Ms.                32860  38955  0.843537
1            Mr.                 8960  10648  0.841473
4        Teacher                 1877   2360  0.795339
0            Dr.                    9     13  0.692308
==================================================
  teacher_prefix  project_is_approved  total       Avg
2           Mrs.                48997  57269  0.855559
3            Ms.                32860  38955  0.843537
1            Mr.                 8960  10648  0.841473
4        Teacher                 1877   2360  0.795339
0            Dr.                    9     13  0.692308
```

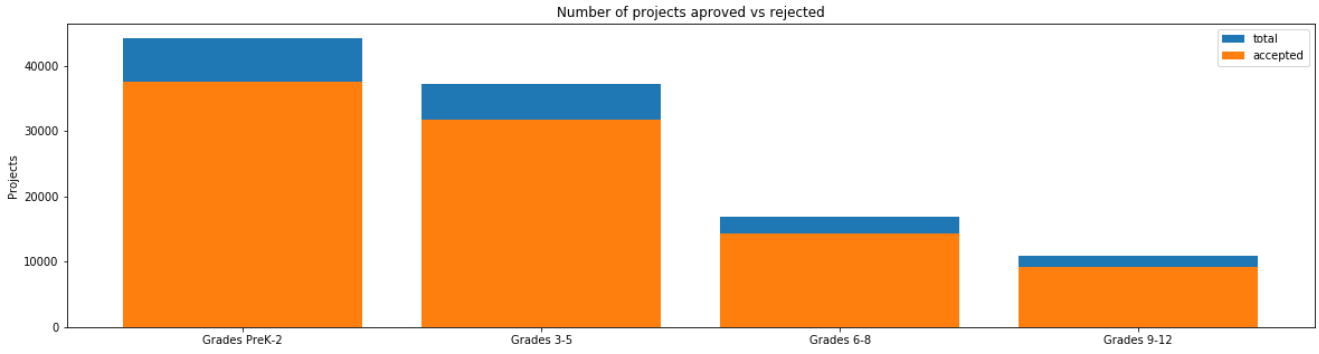
SUMMARY

- Max number of contribution are done by elder teachers (who assumed to be Married), Mrs
- Man (Mr) usually less contribute as compared to Women (Mrs and Ms), this could be because, Man could be less in Teaching/academic field.
- "Teacher" should have be categorized with gender, to get more clarity of number. Though since the number is less (2k) against Mrs,MS and Mr. (57K,38K,10k respectively), it seems, project which are submitted under prefix "Teacher", is because of Group contribution and not a) under single contribution b) not under head by someone else.
- Project submitted by (Mrs,Ms,Mr) has >80% (84 approx) chance to be approved.
- If we safely assume that Dr. are Phd holder(or Lecturer) and not medical practitioner, we can ignore their contribution for Donor Project, as it mere 13. It also says, Phd holder / or senior most teacher (who moved from academic to adimstration side, do less contribution toward Project).

1.2.3 Univariate Analysis: project_grade_category

In [28]:

```
univariate_barplots(project_data, 'project_grade_category', 'project_is_approved', top=False)
```




```
  project_grade_category  project_is_approved  total       Avg
3          Grades PreK-2                37536  44225  0.848751
0             Grades 3-5                31729  37137  0.854377
```

```
1             Grades 6-8                14258  16923  0.842522
2            Grades 9-12                 9183  10963  0.837636
==================================================
  project_grade_category  project_is_approved  total       Avg
3          Grades PreK-2                37536  44225  0.848751
0             Grades 3-5                31729  37137  0.854377
1             Grades 6-8                14258  16923  0.842522
2            Grades 9-12                 9183  10963  0.837636
```

SUMMARY

- Every Grade category has greater than 83% success rate in approval
- Chance of approving of project, is more for the children of lesser age. and with increasing age, chances gets decrease.

1.2.4 Univariate Analysis: project_subject_categories

In [29]:

```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
project_data.head(3)
```

```
Number of data points in train data (109248, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

Out[29]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Gra
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Gra
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	2016-08-31 12:03:56	Gra

In [30]:

```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python:
https://stackoverflow.com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python
cat_list = []
for i in catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care & H
```

```
unger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Scienc
e"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i
.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math &
Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())
```

In [31]:

```
# "project_subject_categories" will be dropped and will replace by "clean_categories"
project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)
project_data.head(2)
```

Out[31]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Gra
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Gra

In [32]:

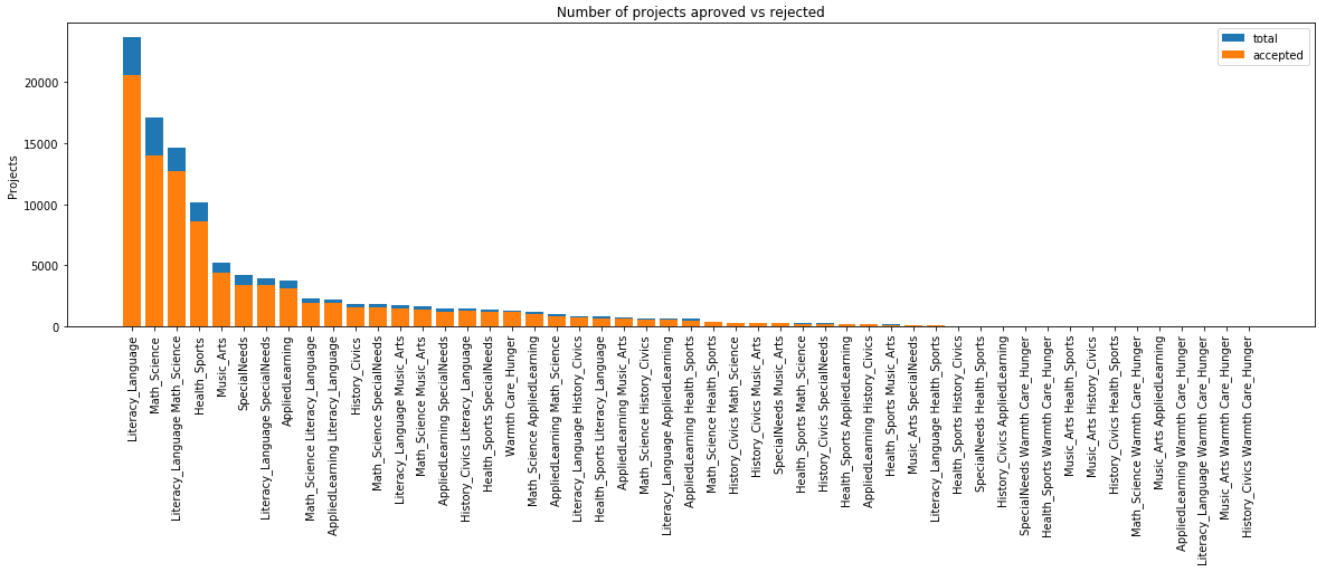
```
#stacked bar plots matplotlib:
https://matplotlib.org/gallery/lines_bars_and_markers/bar_stacked.html

#added Rotation, so to see x axis, by 90 degree
#https://medium.com/python-pandemonium/data-visualization-in-python-bar-graph-in-matplotlib-f17386
02e9c4

def stack_plot(data, xtick, col2='project_is_approved', col3='total'):
    ind = np.arange(data.shape[0])

    #print("="*50)
    #print(data.shape[0])
    #print("="*50)
    # https://medium.com/python-pandemonium/data-visualization-in-python-bar-graph-in-matplotlib-f
1738602e9c4
    # arange is numpymethod that generates an array of sequential numbers. For instance
numpy.arange(5)
    # will generate a numpy.ndArray like [0,1,2,3,4,5]
    # Why is it needed? We need some data for X-axis and right now we only have labels that can't
be used
    # for plotting purpose. So we will generate an array of length of label and use it on X-axis


    plt.figure(figsize=(20,5))
    p1 = plt.bar(ind, data[col3].values) # total
    p2 = plt.bar(ind, data[col2].values) # project is accepted

    plt.ylabel('Projects')
    plt.title('Number of projects aproved vs rejected')
#added Rotation, so to see x axis, by 90 degree
#https://medium.com/python-pandemonium/data-visualization-in-python-bar-graph-in-matplotlib-f17386
02e9c4
    plt.xticks(ind, list(data[xtick].values), rotation=90)
    plt.legend((p1[0], p2[0]), ('total', 'accepted'))
    plt.show()
```

In [33]:

```
def univariate_barplots(data, col1, col2='project_is_approved', top=False):
    # Count number of zeros in dataframe python: https://stackoverflow.com/a/51540521/4084039
    temp = pd.DataFrame(project_data.groupby(col1)[col2].agg(lambda x: x.eq(1).sum())).reset_index(
)

    # Pandas dataframe grouby count: https://stackoverflow.com/a/19385591/4084039
    temp['total'] = pd.DataFrame(project_data.groupby(col1)
[col2].agg({'total':'count'})).reset_index()['total']
    temp['Avg'] = pd.DataFrame(project_data.groupby(col1)[col2].agg({'Avg':'mean'})).reset_index()[
'Avg']

    temp.sort_values(by=['total'],inplace=True, ascending=False)

    #print("="*50)
    #print(temp.head(2))
    #print("="*50)
    if top:
        temp = temp[0:top]

    stack_plot(temp, xtick=col1, col2=col2, col3='total')
    print(temp.head(5))
    print("="*50)
    print(temp.tail(5))

    if col1 == "clean_categories":
        tempAvg=temp
        tempAvg.sort_values(by=['Avg'],inplace=True, ascending=False)
        print("Sort by Average")
        print(tempAvg.head(41)) # trying to get max approval rate
        print("="*50)
        print(tempAvg.tail(12))

    if col1 =="teacher_number_of_previously_posted_projects":
        tempAvg=temp
        tempAvg.sort_values(by=['Avg'],inplace=True, ascending=False)
        print("Sort by Average")
        print(tempAvg.head(5))
        print("="*50)
        print(tempAvg.tail(5))
```

In [34]:

```
# univariate_barplots(project_data, 'clean_categories', 'project_is_approved', top=20)
univariate_barplots(project_data, 'clean_categories', 'project_is_approved', top=False)
```



	clean_categories	project_is_approved	total	Avg
24	Literacy_Language	20520	23655	0.867470
32	Math_Science	13991	17072	0.819529
28	Literacy_Language Math_Science	12725	14636	0.869432
8	Health_Sports	8640	10177	0.848973

```
40                      Music_Arts                 4429   5180  0.855019
==================================================
                       clean_categories  project_is_approved  total       Avg
41           Music_Arts AppliedLearning                    7     10  0.700000
7     AppliedLearning Warmth Care_Hunger                    8     10  0.800000
31  Literacy_Language Warmth Care_Hunger                    7      9  0.777778
45         Music_Arts Warmth Care_Hunger                    1      2  0.500000
23     History_Civics Warmth Care_Hunger                    0      1  0.000000
Sort by Average
                      clean_categories  project_is_approved  total       Avg
15    Health_Sports Warmth Care_Hunger                   22     23  0.956522
50                  Warmth Care_Hunger                 1212   1309  0.925898
18        History_Civics Health_Sports                   12     13  0.923077
19    History_Civics Literacy_Language                 1271   1421  0.894441
10        Health_Sports History_Civics                   38     43  0.883721
27    Literacy_Language History_Civics                  710    809  0.877627
44             Music_Arts SpecialNeeds                  121    138  0.876812
14          Health_Sports SpecialNeeds                 1215   1391  0.873472
28      Literacy_Language Math_Science                12725  14636  0.869432
24                  Literacy_Language                 20520  23655  0.867470
3    AppliedLearning Literacy_Language                 1887   2191  0.861251
36      Math_Science Literacy_Language                 1968   2289  0.859764
20         History_Civics Math_Science                  276    322  0.857143
35         Math_Science History_Civics                  558    652  0.855828
30      Literacy_Language SpecialNeeds                 3389   3961  0.855592
25   Literacy_Language AppliedLearning                  544    636  0.855346
40                         Music_Arts                 4429   5180  0.855019
8                        Health_Sports                8640  10177  0.848973
9         Health_Sports AppliedLearning                  163    192  0.848958
11     Health_Sports Literacy_Language                  679    803  0.845579
29        Literacy_Language Music_Arts                 1475   1757  0.839499
33        Math_Science AppliedLearning                 1019   1220  0.835246
16                      History_Civics                 1545   1851  0.834684
48            SpecialNeeds Music_Arts                  252    302  0.834437
21           History_Civics Music_Arts                 260    312  0.833333
38           Math_Science SpecialNeeds                 1531   1840  0.832065
37             Math_Science Music_Arts                 1366   1642  0.831912
1         AppliedLearning Health_Sports                  501    608  0.824013
12           Health_Sports Math_Science                  223    271  0.822878
2       AppliedLearning History_Civics                  146    178  0.820225
32                        Math_Science                13991  17072  0.819529
0                       AppliedLearning                3072   3771  0.814638
6         AppliedLearning SpecialNeeds                 1195   1467  0.814588
22         History_Civics SpecialNeeds                  205    252  0.813492
4         AppliedLearning Math_Science                  855   1052  0.812738
46                        SpecialNeeds                 3431   4226  0.811879
5           AppliedLearning Music_Arts                  612    758  0.807388
13            Health_Sports Music_Arts                  125    155  0.806452
26     Literacy_Language Health_Sports                   58     72  0.805556
7   AppliedLearning Warmth Care_Hunger                    8     10  0.800000
34          Math_Science Health_Sports                  326    414  0.787440
==================================================
                       clean_categories  project_is_approved  total       Avg
7     AppliedLearning Warmth Care_Hunger                    8     10  0.800000
34           Math_Science Health_Sports                  326    414  0.787440
17        History_Civics AppliedLearning                   33     42  0.785714
47            SpecialNeeds Health_Sports                   33     42  0.785714
49       SpecialNeeds Warmth Care_Hunger                   18     23  0.782609
31  Literacy_Language Warmth Care_Hunger                    7      9  0.777778
43             Music_Arts History_Civics                  13     18  0.722222
41           Music_Arts AppliedLearning                    7     10  0.700000
42             Music_Arts Health_Sports                   13     19  0.684211
39       Math_Science Warmth Care_Hunger                    6     11  0.545455
45         Music_Arts Warmth Care_Hunger                    1      2  0.500000
23     History_Civics Warmth Care_Hunger                    0      1  0.000000
```

SUMMARY

Note: I took all the category, rather than first 20 (this to see the approval rate).

- Literacy_Language and Math_Science or combination of both, are the top three category, against which the Project are Submitted and approve.
- 40 top project for "project_subject_categories" has 80% success rate in approval.
- Warmth and Care_Hunger, together have 1309 project (with 92% approval rate), but if it combine with other (both

individually and together), have very less project submitted (SpecialNeeds Warmth Care_Hunger -23(78%) Literacy_Language Warmth Care_Hunger 9(77%), Math_Science Warmth Care_Hunger 11(54%),Music_Arts Warmth Care_Hunger 2(50%), History_Civics Warmth Care_Hunger 1(0%)

In [35]:

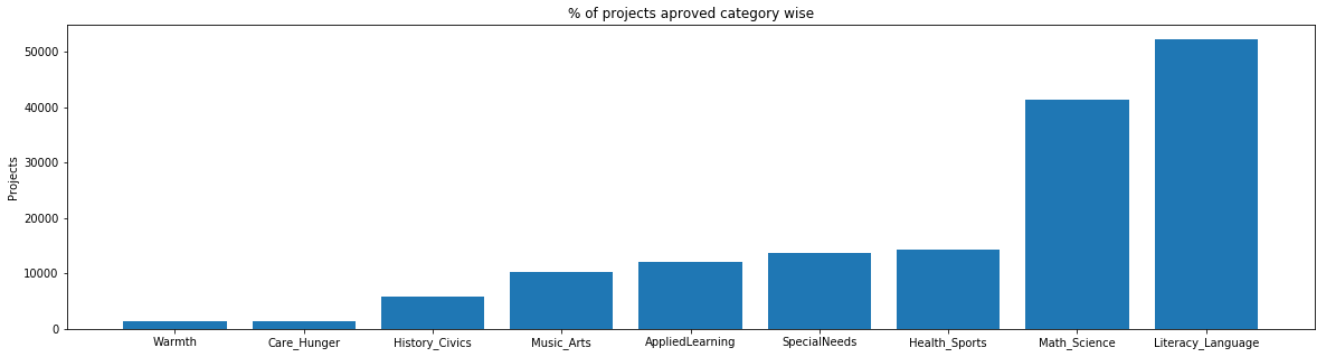
```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
from collections import Counter
my_counter = Counter()
for word in project_data['clean_categories'].values:
    my_counter.update(word.split())

print(my_counter)
```

```
Counter({'Literacy_Language': 52239, 'Math_Science': 41421, 'Health_Sports': 14223,
'SpecialNeeds': 13642, 'AppliedLearning': 12135, 'Music_Arts': 10293, 'History_Civics': 5914,
'Warmth': 1388, 'Care_Hunger': 1388})
```

In [36]:

```
# dict sort by value python: https://stackoverflow.com/a/613218/4084039
cat_dict = dict(my_counter)
sorted_cat_dict = dict(sorted(cat_dict.items(), key=lambda kv: kv[1]))

print(sorted_cat_dict)
ind = np.arange(len(sorted_cat_dict))
print(ind)

plt.figure(figsize=(20,5))
p1 = plt.bar(ind, list(sorted_cat_dict.values()))
print(list(sorted_cat_dict.values()))

plt.ylabel('Projects')
plt.title('% of projects aproved category wise')
plt.xticks(ind, list(sorted_cat_dict.keys()))
plt.show()
```

```
{'Warmth': 1388, 'Care_Hunger': 1388, 'History_Civics': 5914, 'Music_Arts': 10293,
'AppliedLearning': 12135, 'SpecialNeeds': 13642, 'Health_Sports': 14223, 'Math_Science': 41421, 'L
iteracy_Language': 52239}
[0 1 2 3 4 5 6 7 8]
[1388, 1388, 5914, 10293, 12135, 13642, 14223, 41421, 52239]
```



In [37]:

```
for i, j in sorted_cat_dict.items():
    print("{:20} :{:10}".format(i,j))
```

```
Warmth               :      1388
Care_Hunger          :      1388
History_Civics       :      5914
Music_Arts           :     10293
AppliedLearning      :     12135
SpecialNeeds         :     13642
Health_Sports        :     14223
Math_Science         :     41421
Literacy_Language    :     52239
```

Summary: project_subject_categories

- Literacy_Language, Math_Science and Health_Sports are the top three project (52K, 41 K 14K respectively)
- CareHunger, Warmth and HistoryCivics are the bottom three project (5914,1388,1388 respectively)
- Previous Summary's 3rd point and above point infer that Project Category, "CareHunger" and "Warmth" comes together always. either both alone, or alongwith other category.

1.2.5 Univariate Analysis: project_subject_subcategories

In [38]:

```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
project_data.head(2)
```

```
Number of data points in train data (109248, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_subcategories' 'project_title' 'project_essay_1'
 'project_essay_2' 'project_essay_3' 'project_essay_4'
 'project_resource_summary' 'teacher_number_of_previously_posted_projects'
 'project_is_approved' 'clean_categories']
```

Out[38]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Gra
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Gra

In [39]:

```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python:
https://stackoverflow.com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python

sub_cat_list = []
for i in sub_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care & H
unger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Scienc
e"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i
.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math &
Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())
```

In [40]:

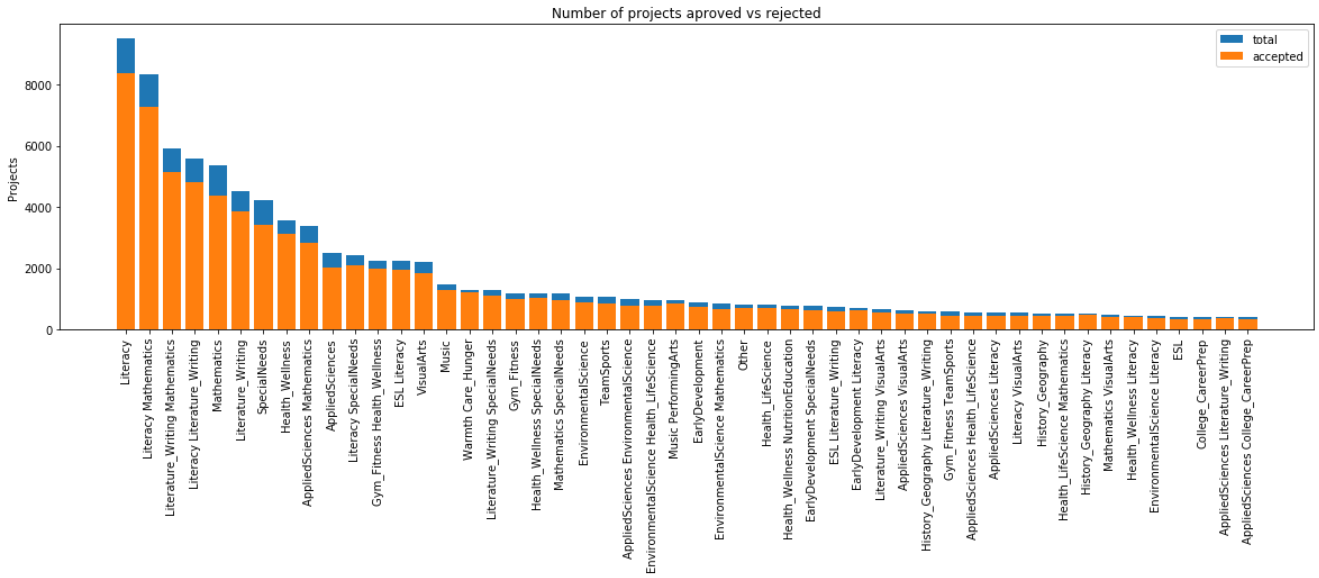
```
# "project_subject_subcategories" will be dropped and will replace by "clean_subcategories"
project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
project_data.head(2)
```

Out[40]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Gra
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Gra

In [41]:

```
univariate_barplots(project_data, 'clean_subcategories', 'project_is_approved', top=50)
```




```
                clean_subcategories  project_is_approved  total       Avg
317                         Literacy                 8371   9486  0.882458
319              Literacy Mathematics                7260   8325  0.872072
331  Literature_Writing Mathematics                  5140   5923  0.867803
318      Literacy Literature_Writing                 4823   5571  0.865733
342                      Mathematics                 4385   5379  0.815207
==================================================
                    clean_subcategories  project_is_approved  total       Avg
196       EnvironmentalScience Literacy                  389    444  0.876126
127                                 ESL                 349    421  0.828979
79                   College_CareerPrep                 343    421  0.814727
17   AppliedSciences Literature_Writing                 361    420  0.859524
3    AppliedSciences College_CareerPrep                 330    405  0.814815
```

SUMMARY : project_subject_subcategories

Sub-category analysis is for top 50 data.

- Top 50 project for "project_subject_subcategories" has more than 81% success rate in approval.
- "Literacy" category has more chances for getting approved.

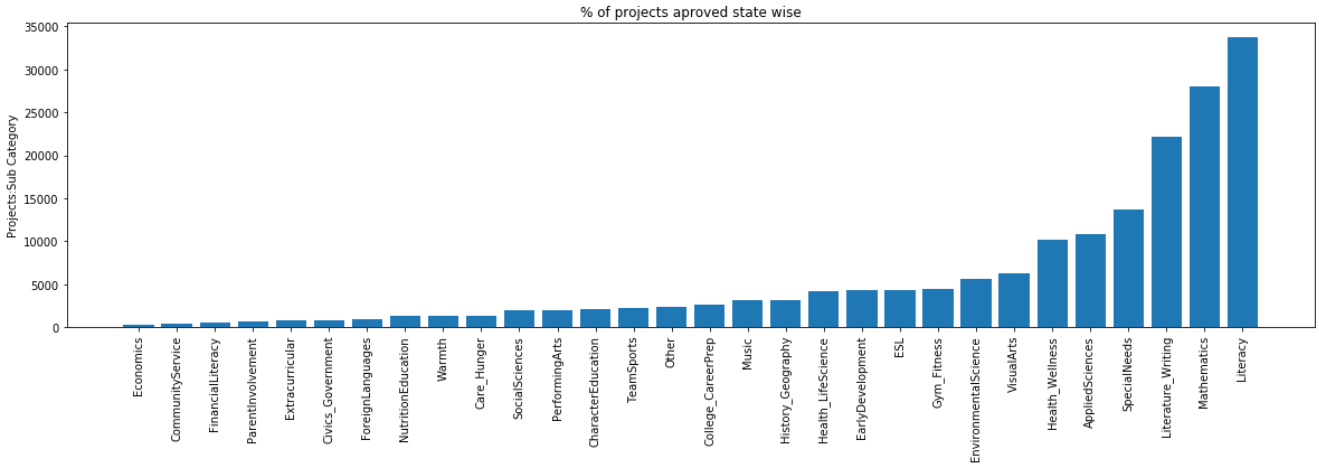
In [42]:

In [42]:

```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
from collections import Counter
my_counter = Counter()
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())
```

In [43]:

```
# dict sort by value python: https://stackoverflow.com/a/613218/4084039
sub_cat_dict = dict(my_counter)
sorted_sub_cat_dict = dict(sorted(sub_cat_dict.items(), key=lambda kv: kv[1]))


ind = np.arange(len(sorted_sub_cat_dict))
plt.figure(figsize=(20,5))
p1 = plt.bar(ind, list(sorted_sub_cat_dict.values()))

plt.ylabel('Projects:Sub Category')
plt.title('% of projects aproved state wise')
#added Rotation, so to see x axis, by 90 degree
#https://medium.com/python-pandemonium/data-visualization-in-python-bar-graph-in-matplotlib-f17386
02e9c4
plt.xticks(ind, list(sorted_sub_cat_dict.keys()), rotation=90)
plt.show()
```



In [44]:

```
for i, j in sorted_sub_cat_dict.items():
    print("{:20} :{:10}".format(i,j))
```

```
Economics            :       269
CommunityService     :       441
FinancialLiteracy    :       568
ParentInvolvement    :       677
Extracurricular      :       810
Civics_Government    :       815
ForeignLanguages     :       890
NutritionEducation   :      1355
Warmth               :      1388
Care_Hunger          :      1388
SocialSciences       :      1920
PerformingArts       :      1961
CharacterEducation   :      2065
TeamSports           :      2192
Other                :      2372
College_CareerPrep   :      2568
Music                :      3145
History_Geography    :      3171
Health_LifeScience   :      4235
EarlyDevelopment     :      4254
ESL                  :      4367
Gym_Fitness          :      4509
EnvironmentalScience :      5591
VisualArts           :      6278
```

```
Health_Wellness      :     10234
AppliedSciences      :     10816
SpecialNeeds         :     13642
Literature_Writing   :     22179
Mathematics          :     28074
Literacy             :     33700
```

Summary: project_subject_subcategories

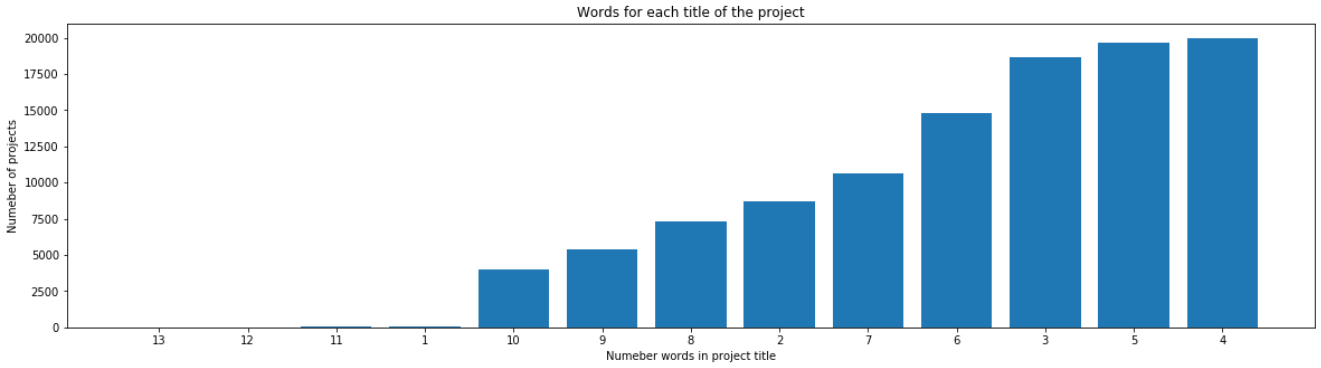
- Literacy, Mathematics, Literature_Writing are the top three project (33K, 28K, 22K respectively)
- Economics, CommunityService, and FinancialLiteracy are the bottom three project (269, 441,568 respectively)

1.2.6 Univariate Analysis: Text features (Title)

In [45]:

```
#How to calculate number of words in a string in DataFrame:
https://stackoverflow.com/a/37483537/4084039
print(type(project_data['project_title']))
print(type(project_data['project_title']).str)
word_count = project_data['project_title'].str.split().apply(len).value_counts()
print(word_count)
word_dict = dict(word_count)
print(word_dict)
word_dict = dict(sorted(word_dict.items(), key=lambda kv: kv[1]))
print(word_dict)

ind = np.arange(len(word_dict))
plt.figure(figsize=(20,5))
p1 = plt.bar(ind, list(word_dict.values()))

plt.ylabel('Numeber of projects')
plt.xlabel('Numeber words in project title')
plt.title('Words for each title of the project')
plt.xticks(ind, list(word_dict.keys()))
plt.show()
```

```
<class 'pandas.core.series.Series'>
<class 'pandas.core.strings.StringMethods'>
4     19979
5     19677
3     18691
6     14824
7     10631
2      8733
8      7289
9      5383
10     3968
1        31
11       30
12       11
13        1
Name: project_title, dtype: int64
{4: 19979, 5: 19677, 3: 18691, 6: 14824, 7: 10631, 2: 8733, 8: 7289, 9: 5383, 10: 3968, 1: 31, 11:
30, 12: 11, 13: 1}
{13: 1, 12: 11, 11: 30, 1: 31, 10: 3968, 9: 5383, 8: 7289, 2: 8733, 7: 10631, 6: 14824, 3: 18691,
5: 19677, 4: 19979}
```



Summary : Project Title

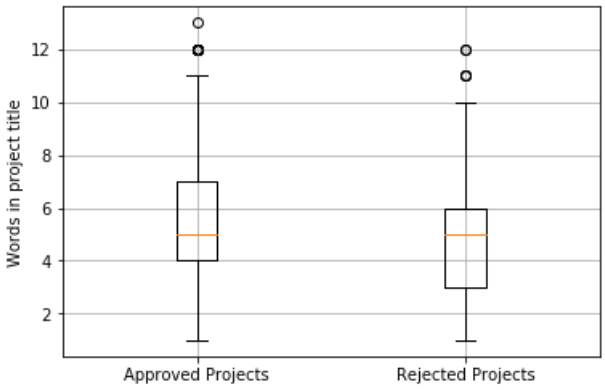
- Fewer the words are found more in "Project title" submitted (except 1)
- Double digit (except 10), seldomly appears in Projects title.

In [47]:

```
approved_title_word_count = project_data[project_data['project_is_approved']==1]['project_title'].
str.split().apply(len)
approved_title_word_count = approved_title_word_count.values

rejected_title_word_count = project_data[project_data['project_is_approved']==0]['project_title'].
str.split().apply(len)
rejected_title_word_count = rejected_title_word_count.values
```

In [48]:

```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved_title_word_count, rejected_title_word_count])
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Words in project title')
plt.grid()
plt.show()
```

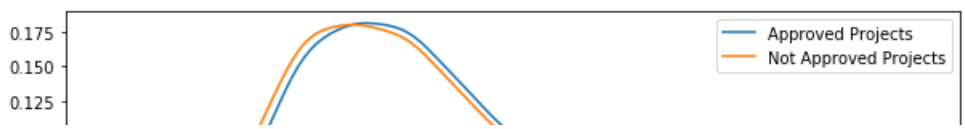


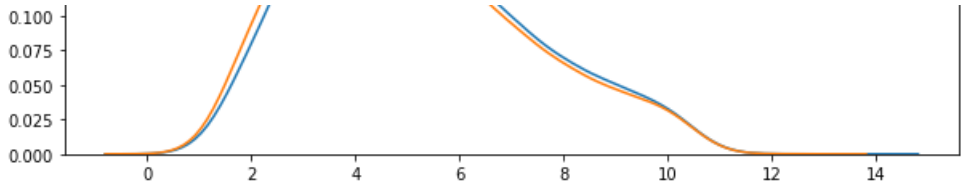
Summary: project title

- Median of both Aproved and Rejected project are almost same.
- Distinctively we can't say, that number of word count in project title, will result in Approval or rejection of the Project

In [49]:

```
plt.figure(figsize=(10,3))

# https://seaborn.pydata.org/generated/seaborn.kdeplot.html | kernel density estimate | sns.kdeplo
t
# bw : {'scott' | 'silverman' | scalar | pair of scalars }, optional
# Name of reference method to determine kernel size, scalar factor, or scalar for each dimension
# of the bivariate plot. Note that the underlying computational libraries have different
interperetations
# for this parameter: statsmodels uses it directly, but scipy treats it as a scaling factor
# for the standard deviation of the data.

sns.kdeplot(approved_title_word_count,label="Approved Projects", bw=0.6)
sns.kdeplot(rejected_title_word_count,label="Not Approved Projects", bw=0.6)
plt.legend()
plt.show()

#PDF
```





Summary

- Till the word 4(blue and orange line intersect), number of project, rejected more(orange), against the project approved(blue), for a given words
- after 4 (blue and orange line intersect), Approve project are more than rejected Project, for a no of work count, until it reaches 10 (word count), for which, approve and rejection are same)
- for one word in project title, no of approval and rejection are same/similiar.

1.2.7 Univariate Analysis: Text features (Project Essay's)

In [50]:

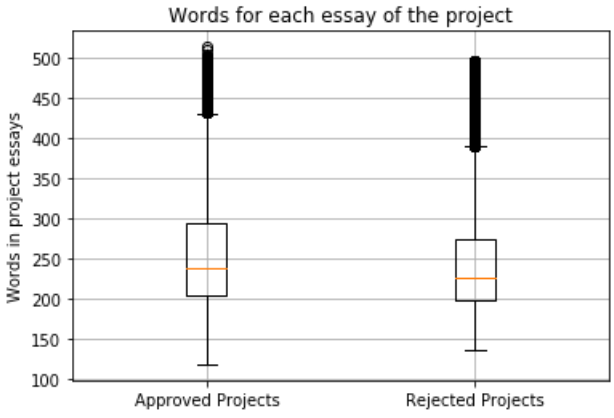
```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)
```

In [51]:

```
approved_word_count = project_data[project_data['project_is_approved']==1]['essay'].str.split().apply(len)
approved_word_count = approved_word_count.values

rejected_word_count = project_data[project_data['project_is_approved']==0]['essay'].str.split().apply(len)
rejected_word_count = rejected_word_count.values
```

In [52]:

```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved_word_count, rejected_word_count])
plt.title('Words for each essay of the project')
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Words in project essays')
plt.grid()
plt.show()
```

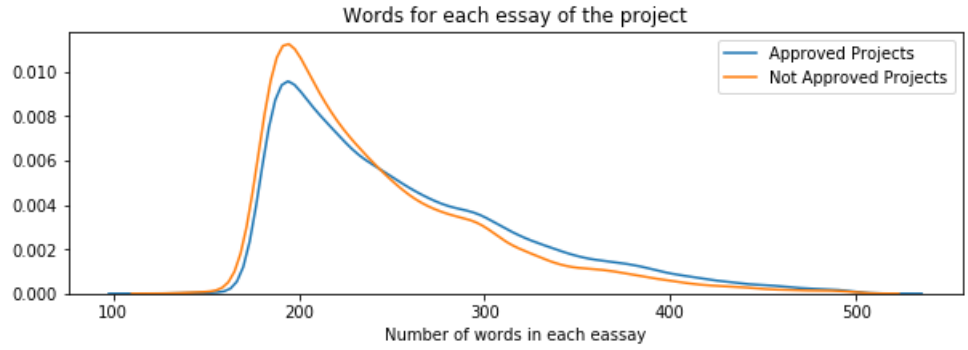


Summary

- Median of approved Project is more (same as 75 percentile) than the rejected Project (and its 75 percentile), it shows that definately more words (essay description), help to get the project approve.

In [53]:

```
# sns.distplot | https://seaborn.pydata.org/generated/seaborn.distplot.html

plt.figure(figsize=(10,3))
sns.distplot(approved_word_count, hist=False, label="Approved Projects")
sns.distplot(rejected_word_count, hist=False, label="Not Approved Projects")
plt.title('Words for each essay of the project')
plt.xlabel('Number of words in each eassay')
plt.legend()
plt.show()
```



Summary

- Somewhere between 240 words (blue and orange intersect), shows, project got rejected, for less than 240 words
- for more than 250 words, chances of Project acceptance is more, until 500 words, where both rejectrion and approval are similiar.

1.2.8 Univariate Analysis: Cost per project

In [54]:

```
# we get the cost of the project using resource.csv file
resource_data.head(2)
```

Out[54]:

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95

In [55]:

```
# https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-indexes-for-all-groups-in-one-step
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
price_data.head(2)
```

Out[55]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [56]:

```
# join two dataframes in python:
# https://www.shanelynn.ie/merge-join-dataframes-python-pandas-index-1/

project_data = pd.merge(project_data, price_data, on='id', how='left')
```

```
print(project_data.shape)
project_data.head(4)
```

```
(109248, 20)
```

Out[56]:

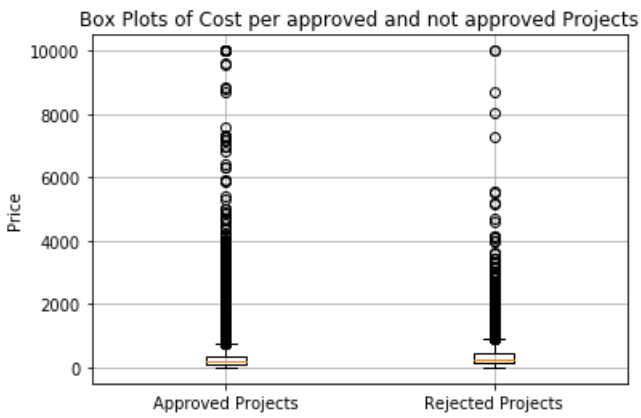
	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Gra
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Gra
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	2016-08-31 12:03:56	Gra
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	2016-10-06 21:16:17	Gra

In [57]:

```
approved_price = project_data[project_data['project_is_approved']==1]['price'].values

rejected_price = project_data[project_data['project_is_approved']==0]['price'].values
```

In [58]:

```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved_price, rejected_price])
plt.title('Box Plots of Cost per approved and not approved Projects')
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Price')
plt.grid()
plt.show()
```

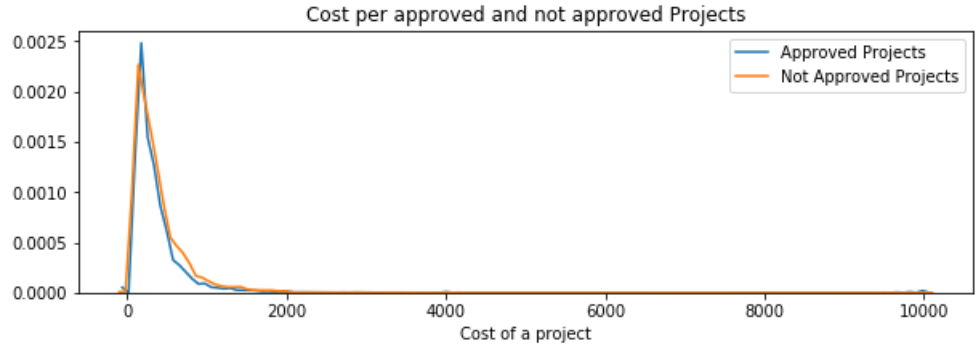


Summary

Cannot conclude anything, since all the percentile, are overlapping.

In [59]:

```
plt.figure(figsize=(10,3))
sns.distplot(approved_price, hist=False, label="Approved Projects")
sns.distplot(rejected_price, hist=False, label="Not Approved Projects")
plt.title('Cost per approved and not approved Projects')
plt.xlabel('Cost of a project')
plt.legend()
plt.show()
```



Summary

- cannot conclude much, since lines are overlapping

In [48]:

```
# http://zetcode.com/python/prettytable/
from prettytable import PrettyTable

#If you get a ModuleNotFoundError error , install prettytable using: pip3 install prettytable

x = PrettyTable()
x.field_names = ["Percentile", "Approved Projects", "Not Approved Projects"]

for i in range(0,101,5):
    x.add_row([i,np.round(np.percentile(approved_price,i), 3), np.round(np.percentile(rejected_pric
e,i), 3)])
print(x)
```

```
+------------+-------------------+------------------------+
| Percentile | Approved Projects | Not Approved Projects |
+------------+-------------------+------------------------+
|      0     |        0.66       |          1.97          |
|      5     |       13.59       |          41.9          |
|     10     |       33.88       |         73.67          |
|     15     |        58.0       |         99.109         |
|     20     |       77.38       |         118.56         |
|     25     |       99.95       |        140.892         |
|     30     |       116.68      |         162.23         |
|     35     |      137.232      |        184.014         |
|     40     |       157.0       |        208.632         |
|     45     |      178.265      |        235.106         |
|     50     |       198.99      |        263.145         |
|     55     |       223.99      |         292.61         |
|     60     |       255.63      |        325.144         |
|     65     |      285.412      |         362.39         |
|     70     |      321.225      |         399.99         |
|     75     |      366.075      |        449.945         |
|     80     |       411.67      |        519.282         |
|     85     |       479.0       |        618.276         |
|     90     |       593.11      |        739.356         |
```

```
|     95     |      801.598       |        992.486         |
|    100     |       9999.0       |         9999.0         |
+------------+--------------------+------------------------+
```

Summary

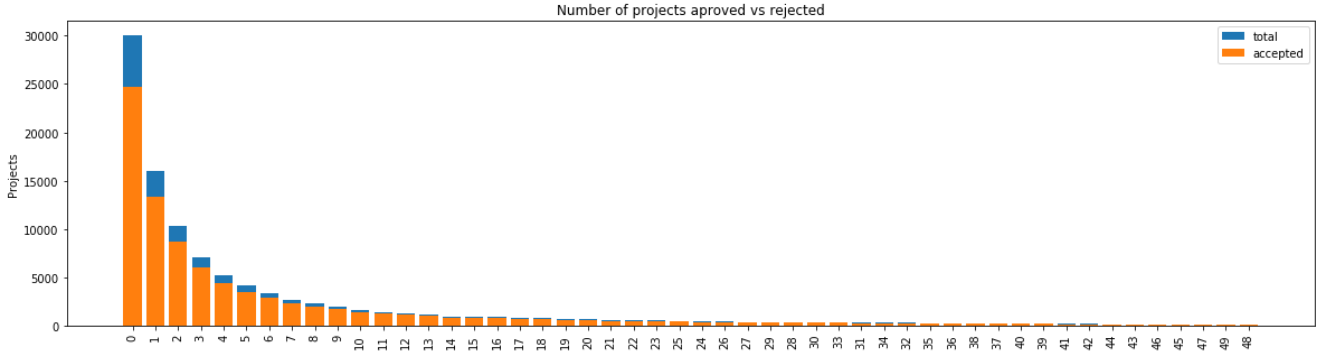
Cannot conclude anything, but approved project has less amount as compare to the rejected project, agasint each percentile.

1.2.9 Univariate Analysis: teacher_number_of_previously_posted_projects

Please do this on your own based on the data analysis that was done in the above cells

In [60]:

```
# https://stackoverflow.com/a/19385591/4084039
# https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-indexes-for-all-groups-in-one-step
# https://www.shanelynn.ie/summarising-aggregation-and-grouping-data-in-python-pandas/
univariate_barplots(project_data, 'teacher_number_of_previously_posted_projects',
'project_is_approved', top=50)
```




```
   teacher_number_of_previously_posted_projects  project_is_approved  total  \
0                                             0                24652  30014
1                                             1                13329  16058
2                                             2                 8705  10350
3                                             3                 5997   7110
4                                             4                 4452   5266

        Avg
0  0.821350
1  0.830054
2  0.841063
3  0.843460
4  0.845423
==================================================
    teacher_number_of_previously_posted_projects  project_is_approved  total  \
46                                            46                 149    164
45                                            45                 141    153
47                                            47                 129    144
49                                            49                 128    143
48                                            48                 135    140

         Avg
46  0.908537
45  0.921569
47  0.895833
49  0.895105
48  0.964286
Sort by Average
    teacher_number_of_previously_posted_projects  project_is_approved  total  \
48                                            48                 135    140
45                                            45                 141    153
32                                            32                 276    301
40                                            40                 202    221
38                                            38                 224    246

         Avg
48  0.964286
```

```
46  0.964286
45  0.921569
32  0.916944
40  0.914027
38  0.910569
==================================================
   teacher_number_of_previously_posted_projects  project_is_approved  total  \
4                                             4                4452   5266
3                                             3                5997   7110
2                                             2                8705  10350
1                                             1               13329  16058
0                                             0               24652  30014

        Avg
4  0.845423
3  0.843460
2  0.841063
1  0.830054
0  0.821350
```

Summary

Data taken for top 50 record

- First time submission are maximum(30k), and they are also approved 82 times (24k).
- Gradually, submission of project decreases.
- Seldomly, people make multpli submission. For ex. 48 project are submitted by 140 people only.
- If their were previous submision, the more the approval rate is. For ex. Highest approval rate (96%) is for the 48 project submission.

  All in all, submission approval rate is above 82% for top 50 project.

1.2.10 Univariate Analysis: project_resource_summary

Please do this on your own based on the data analysis that was done in the above cells

Check if the `presence of the numerical digits` in the `project_resource_summary` effects the acceptance of the project or not. If you observe that `presence of the numerical digits` is helpful in the classification, please include it for further process or you can ignore it.

In [61]:

```
# how to detech digit in a string in python | https://stackoverflow.com/questions/19859282/check-if-a-string-contains-a-number
def hasNumbers(inputString):
    return any(char.isdigit() for char in inputString)

prjSumm = list(project_data['project_resource_summary'].values)
#print(type(prjSumm))
prjSum_list = []
#print(type(prjSum_list))

# used alteration code of 'clean category'
for inputString in prjSumm:
    #temp = ""
    if hasNumbers(inputString):
        prjSum_list.append(1) #prjSum_list=1
    else:
        prjSum_list.append(0)

# print(prjSum_list)
project_data['Prj_Res_Summary_IsDigit'] = prjSum_list
project_data.head(2)

digitPresent = project_data[project_data['Prj_Res_Summary_IsDigit']==1]
digitPresent.head(2)
#approved_title_word_count = project_data[project_data['project_is_approved']==1]
['project_title'].str.split().apply(len)
#approved_title_word_count = approved_title_word_count.values
```

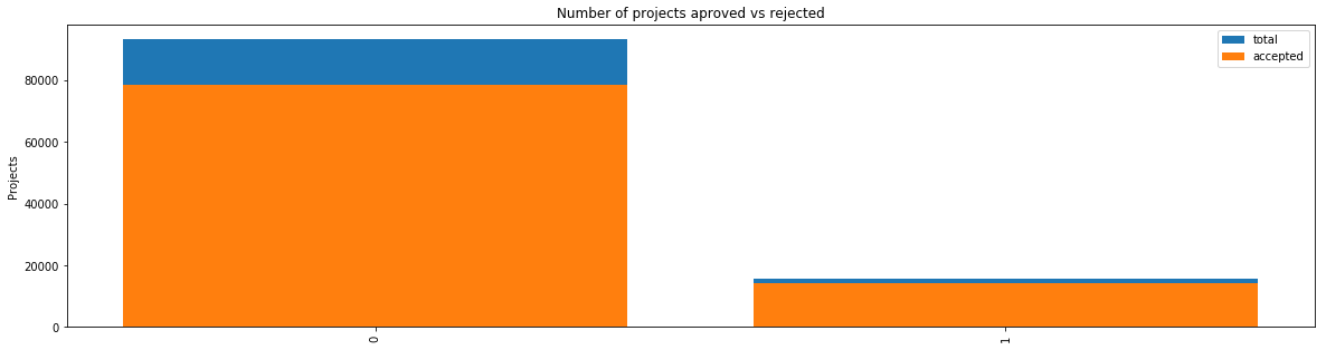
Out[61]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pr
12	19090	p051126	5e52c92b7e3c472aad247a239d345543	Mrs.	NY	2016-05-23 15:46:02	G
14	62232	p233127	424819801de22a60bba7d0f4354d0258	Ms.	MA	2017-02-14 16:29:10	G

2 rows × 21 columns

In [62]:

```
univariate_barplots(project_data, 'Prj_Res_Summary_IsDigit', 'project_is_approved', top=50)
```




```
   Prj_Res_Summary_IsDigit  project_is_approved  total       Avg
0                        0                78616  93492  0.840885
1                        1                14090  15756  0.894263
==================================================
   Prj_Res_Summary_IsDigit  project_is_approved  total       Avg
0                        0                78616  93492  0.840885
1                        1                14090  15756  0.894263
```

Summary

- Surely, digit in 'project resource summary' give the edge against the non- numeric text, for getting approval (89% against 84%), however the total data point are very less (95K against 15K) to anslysis.

1.3 Text preprocessing

1.3.1 Essay Text

In [63]:

```
project_data.head(2)
```

Out[63]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Gra
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Gra

2 rows × 21 columns

In [54]:

```
# printing some random essays.
print(project_data['essay'].values[0])
print("="*50)
print(project_data['essay'].values[150])
print("="*50)
print(project_data['essay'].values[1000])
print("="*50)
print(project_data['essay'].values[20000])
print("="*50)
print(project_data['essay'].values[99999])
print("="*50)
```

```
My students are English learners that are working on English as their second or third languages. W
e are a melting pot of refugees, immigrants, and native-born Americans bringing the gift of langua
ge to our school. \r\n\r\n We have over 24 languages represented in our English Learner program wi
th students at every level of mastery.  We also have over 40 countries represented with the
families within our school.  Each student brings a wealth of knowledge and experiences to us that
open our eyes to new cultures, beliefs, and respect.\"The limits of your language are the limits o
f your world.\"-Ludwig Wittgenstein  Our English learner's have a strong support system at home th
at begs for more resources.  Many times our parents are learning to read and speak English along s
ide of their children.  Sometimes this creates barriers for parents to be able to help their child
learn phonetics, letter recognition, and other reading skills.\r\n\r\nBy providing these dvd's and
players, students are able to continue their mastery of the English language even if no one at hom
e is able to assist.  All families with students within the Level 1 proficiency status, will be a
offered to be a part of this program.  These educational videos will be specially chosen by the En
glish Learner Teacher and will be sent home regularly to watch.  The videos are to help the child
develop early reading skills.\r\n\r\nParents that do not have access to a dvd player will have the
opportunity to check out a dvd player to use for the year.  The plan is to use these videos and ed
ucational dvd's for the years to come for other EL students.\r\nnannan
==================================================
The 51 fifth grade students that will cycle through my classroom this year all love learning, at l
east most of the time. At our school, 97.3% of the students receive free or reduced price lunch. O
f the 560 students, 97.3% are minority students. \r\nThe school has a vibrant community that loves
to get together and celebrate. Around Halloween there is a whole school parade to show off the bea
utiful costumes that students wear. On Cinco de Mayo we put on a big festival with crafts made by
the students, dances, and games. At the end of the year the school hosts a carnival to celebrate t
he hard work put in during the school year, with a dunk tank being the most popular activity.My st
udents will use these five brightly colored Hokki stools in place of regular, stationary, 4-legged
chairs. As I will only have a total of ten in the classroom and not enough for each student to hav
e an individual one, they will be used in a variety of ways. During independent reading time they
will be used as special chairs students will each use on occasion. I will utilize them in place of
chairs at my small group tables during math and reading times. The rest of the day they will be us
ed by the students who need the highest amount of movement in their life in order to stay focused
on school.\r\n\r\nWhenever asked what the classroom is missing, my students always say more Hokki
Stools. They can't get their fill of the 5 stools we already have. When the students are sitting i
n group with me on the Hokki Stools, they are always moving, but at the same time doing their
work. Anytime the students get to pick where they can sit, the Hokki Stools are the first to be ta
ken. There are always students who head over to the kidney table to get one of the stools who are
disappointed as there are not enough of them. \r\n\r\nWe ask a lot of students to sit for 7 hours
a day. The Hokki stools will be a compromise that allow my students to do desk work and move at th
e same time. These stools will help students to meet their 60 minutes a day of movement by
allowing them to activate their core muscles for balance while they sit. For many of my students,
these chairs will take away the barrier that exists in schools for a child who can't sit
still.nannan
```

```
==================================================
How do you remember your days of school? Was it in a sterile environment with plain walls, rows of desks, and a teacher in front of the room? A typical day in our room is nothing like that. I work hard to create a warm inviting themed room for my students look forward to coming to each day.\r\n\r\nMy class is made up of 28 wonderfully unique boys and girls of mixed races in Arkansas.\r\nThey attend a Title I school, which means there is a high enough percentage of free and reduced-price lunch to qualify. Our school is an \"open classroom\" concept, which is very unique as there are no walls separating the classrooms. These 9 and 10 year-old students are very eager learners; they are like sponges, absorbing all the information and experiences and keep on wanting more.With these resources such as the comfy red throw pillows and the whimsical nautical hanging decor and the blue fish nets, I will be able to help create the mood in our classroom setting to be one of a themed nautical environment. Creating a classroom environment is very important in the success in each and every child's education. The nautical photo props will be used with each child as they step foot into our classroom for the first time on Meet the Teacher evening. I'll take pictures of each child with them, have them developed, and then hung in our classroom ready for their first day of 4th grade.  This kind gesture will set the tone before even the first day of school! The nautical thank you cards will be used throughout the year by the students as they create thank you cards to their team groups.\r\n\r\nYour generous donations will help me to help make our classroom a fun, inviting, learning environment from day one.\r\n\r\nIt costs lost of money out of my own pocket on resources to get our classroom ready. Please consider helping with this project to make our new school year a very successful one. Thank you!nannan
==================================================
My kindergarten students have varied disabilities ranging from speech and language delays, cognitive delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work their hardest working past their limitations. \r\n\r\nThe materials we have are the ones I seek out for my students. I teach in a Title I school where most of the students receive free or reduced price lunch.  Despite their disabilities and limitations, my students love coming to school and come eager to learn and explore.Have you ever felt like you had ants in your pants and you needed to groove and move as you were in a meeting? This is how my kids feel all the time. The want to be able to move as they learn or so they say.Wobble chairs are the answer and I love then because they develop their core, which enhances gross motor and in Turn fine motor skills. \r\nThey also want to learn through games, my kids don't want to sit and do worksheets. They want to learn to count by jumping and playing. Physical engagement is the key to our success. The number toss and color and shape mats can make that happen. My students will forget they are doing work and just have the fun a 6 year old deserves.nannan
==================================================
The mediocre teacher tells. The good teacher explains. The superior teacher demonstrates. The great teacher inspires. -William A. Ward\r\n\r\nMy school has 803 students which is makeup is 97.6% African-American, making up the largest segment of the student body. A typical school in Dallas is made up of 23.2% African-American students. Most of the students are on free or reduced lunch. We aren't receiving doctors, lawyers, or engineers children from rich backgrounds or neighborhoods. As an educator I am inspiring minds of young children and we focus not only on academics but one smart, effective, efficient, and disciplined students with good character.In our classroom we can utilize the Bluetooth for swift transitions during class. I use a speaker which doesn't amplify the sound enough to receive the message. Due to the volume of my speaker my students can't hear videos or books clearly and it isn't making the lessons as meaningful. But with the bluetooth speaker my students will be able to hear and I can stop, pause and replay it at any time.\r\nThe cart will allow me to have more room for storage of things that are needed for the day and has an extra part to it I can use.  The table top chart has all of the letter, words and pictures for students to learn about different letters and it is more accessible.nannan
==================================================
```

In [64]:

```
# https://stackoverflow.com/a/47091490/4084039
import re

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
```

In [65]:

```
sent = decontracted(project_data['essay'].values[20000])
print(sent)
print("="*50)
```

```
My kindergarten students have varied disabilities ranging from speech and language delays, cogniti
ve delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work th
eir hardest working past their limitations. \r\n\r\nThe materials we have are the ones I seek out
for my students. I teach in a Title I school where most of the students receive free or reduced pr
ice lunch.  Despite their disabilities and limitations, my students love coming to school and come
eager to learn and explore.Have you ever felt like you had ants in your pants and you needed to gr
oove and move as you were in a meeting? This is how my kids feel all the time. The want to be able
to move as they learn or so they say.Wobble chairs are the answer and I love then because they dev
elop their core, which enhances gross motor and in Turn fine motor skills. \r\nThey also want to l
earn through games, my kids do not want to sit and do worksheets. They want to learn to count by j
umping and playing. Physical engagement is the key to our success. The number toss and color and s
hape mats can make that happen. My students will forget they are doing work and just have the fun
a 6 year old deserves.nannan
==================================================
```

In [66]:

```
# \r \n \t remove from string python: http://texthandler.com/info/remove-line-breaks-python/
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
My kindergarten students have varied disabilities ranging from speech and language delays, cogniti
ve delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work th
eir hardest working past their limitations.     The materials we have are the ones I seek out for
my students. I teach in a Title I school where most of the students receive free or reduced price
lunch.  Despite their disabilities and limitations, my students love coming to school and come eag
er to learn and explore.Have you ever felt like you had ants in your pants and you needed to groov
e and move as you were in a meeting? This is how my kids feel all the time. The want to be able to
move as they learn or so they say.Wobble chairs are the answer and I love then because they develo
p their core, which enhances gross motor and in Turn fine motor skills.   They also want to learn t
hrough games, my kids do not want to sit and do worksheets. They want to learn to count by jumping
and playing. Physical engagement is the key to our success. The number toss and color and shape ma
ts can make that happen. My students will forget they are doing work and just have the fun a 6 yea
r old deserves.nannan
```

In [67]:

```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
My kindergarten students have varied disabilities ranging from speech and language delays cognitiv
e delays gross fine motor delays to autism They are eager beavers and always strive to work their
hardest working past their limitations The materials we have are the ones I seek out for my studen
ts I teach in a Title I school where most of the students receive free or reduced price lunch
Despite their disabilities and limitations my students love coming to school and come eager to lea
rn and explore Have you ever felt like you had ants in your pants and you needed to groove and mov
e as you were in a meeting This is how my kids feel all the time The want to be able to move as th
ey learn or so they say Wobble chairs are the answer and I love then because they develop their co
re which enhances gross motor and in Turn fine motor skills They also want to learn through games
my kids do not want to sit and do worksheets They want to learn to count by jumping and playing Ph
ysical engagement is the key to our success The number toss and color and shape mats can make that
happen My students will forget they are doing work and just have the fun a 6 year old deserves nan
nan
```

In [68]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",
\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his',
'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them',
'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll",
'these', 'those', \
```

```
'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having',
'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', '
while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during',
'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under'
, 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'e
ach', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll'
, 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "d
esn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
"mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn',
"wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [69]:

```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 109248/109248
[01:06<00:00, 1652.76it/s]
```

In [70]:

```
# after preprocesing
preprocessed_essays[20000]
```

Out[70]:

```
'my kindergarten students varied disabilities ranging speech language delays cognitive delays gros
s fine motor delays autism they eager beavers always strive work hardest working past limitations
the materials ones i seek students i teach title i school students receive free reduced price lunc
h despite disabilities limitations students love coming school come eager learn explore have ever
felt like ants pants needed groove move meeting this kids feel time the want able move learn say w
obble chairs answer i love develop core enhances gross motor turn fine motor skills they also want
learn games kids not want sit worksheets they want learn count jumping playing physical engagement
key success the number toss color shape mats make happen my students forget work fun 6 year old de
serves nannan'
```

1.3.2 Project title Text

In [0]:

```
# similarly you can preprocess the titles also
```

In [71]:

```
project_data.head(2)
```

Out[71]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Gra
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Gra

2 rows × 21 columns

In [72]:

```
# printing some random essays.
print(project_data['project_title'].values[0])
print("="*50)
print(project_data['project_title'].values[150])
print("="*50)
print(project_data['project_title'].values[1000])
print("="*50)
print(project_data['project_title'].values[20000])
print("="*50)
print(project_data['project_title'].values[99999])
print("="*50)
```

```
Educational Support for English Learners at Home
==================================================
More Movement with Hokki Stools
==================================================
Sailing Into a Super 4th Grade Year
==================================================
We Need To Move It While We Input It!
==================================================
Inspiring Minds by Enhancing the Educational Experience
==================================================
```

In [73]:

```
sent_title = decontracted(project_data['project_title'].values[20000])
print(sent_title)
print("="*50)
```

```
We Need To Move It While We Input It!
==================================================
```

In [74]:

```
# \r \n \t remove from string python: http://texthandler.com/info/remove-line-breaks-python/
sent_title = sent_title.replace('\\r', ' ')
sent_title = sent_title.replace('\\"', ' ')
sent_title = sent_title.replace('\\n', ' ')
print(sent_title)
```

```
We Need To Move It While We Input It!
```

In [75]:

```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent_title = re.sub('[^A-Za-z0-9]+', ' ', sent_title)
```

```
print(sent_title)
```

```
We Need To Move It While We Input It
```

In [76]:

```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_title = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['project_title'].values):
    sent_title = decontracted(sentance)
    sent_title = sent_title.replace('\\r', ' ')
    sent_title = sent_title.replace('\\"', ' ')
    sent_title = sent_title.replace('\\n', ' ')
    sent_title = re.sub('[^A-Za-z0-9]+', ' ', sent_title)
    # https://gist.github.com/sebleier/554280
    sent_title = ' '.join(e for e in sent_title.split() if e not in stopwords)
    preprocessed_title.append(sent_title.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████| 109248/109248
[00:02<00:00, 36690.86it/s]
```

In [77]:

```
# after preprocesing
preprocessed_title[10]
```

Out[77]:

```
'reading changes lives'
```

1. 4 Preparing data for models

In [78]:

```
project_data.columns
```

Out[78]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', 'project_title',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'essay', 'price', 'quantity',
       'Prj_Res_Summary_IsDigit'],
      dtype='object')
```

we are going to consider

- school_state : categorical data
- clean_categories : categorical data
- clean_subcategories : categorical data
- project_grade_category : categorical data
- teacher_prefix : categorical data

- project_title : text data
- text : text data
- project_resource_summary: text data

- quantity : numerical
- teacher_number_of_previously_posted_projects : numerical
- price : numerical

1.4.1 Vectorizing Categorical data

1.4.1 Vectorizing Categorical data

- https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/

school_state : categorical data (one hot encoding)

In [79]:

```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
from collections import Counter
my_counter_scl_st = Counter()
for word in project_data['school_state'].values:
    my_counter_scl_st.update(word.split())

print(my_counter_scl_st)
```

```
Counter({'CA': 15388, 'TX': 7396, 'NY': 7318, 'FL': 6185, 'NC': 5091, 'IL': 4350, 'GA': 3963, 'SC':
3936, 'MI': 3161, 'PA': 3109, 'IN': 2620, 'MO': 2576, 'OH': 2467, 'LA': 2394, 'MA': 2389, 'WA': 233
4, 'OK': 2276, 'NJ': 2237, 'AZ': 2147, 'VA': 2045, 'WI': 1827, 'AL': 1762, 'UT': 1731, 'TN': 1688,
'CT': 1663, 'MD': 1514, 'NV': 1367, 'MS': 1323, 'KY': 1304, 'OR': 1242, 'MN': 1208, 'CO': 1111, 'AR
': 1049, 'ID': 693, 'IA': 666, 'KS': 634, 'NM': 557, 'DC': 516, 'HI': 507, 'ME': 505, 'WV': 503, 'NH
': 348, 'AK': 345, 'DE': 343, 'NE': 309, 'SD': 300, 'RI': 285, 'MT': 245, 'ND': 143, 'WY': 98,
'VT': 80})
```

In [80]:

```
scl_st_dict = dict(my_counter_scl_st)
sorted_scl_st_dict = dict(sorted(scl_st_dict.items(), key=lambda kv: kv[1]))

print(sorted_scl_st_dict)
```

```
{'VT': 80, 'WY': 98, 'ND': 143, 'MT': 245, 'RI': 285, 'SD': 300, 'NE': 309, 'DE': 343, 'AK': 345,
'NH': 348, 'WV': 503, 'ME': 505, 'HI': 507, 'DC': 516, 'NM': 557, 'KS': 634, 'IA': 666, 'ID': 693,
'AR': 1049, 'CO': 1111, 'MN': 1208, 'OR': 1242, 'KY': 1304, 'MS': 1323, 'NV': 1367, 'MD': 1514, 'CT
': 1663, 'TN': 1688, 'UT': 1731, 'AL': 1762, 'WI': 1827, 'VA': 2045, 'AZ': 2147, 'NJ': 2237, 'OK':
2276, 'WA': 2334, 'MA': 2389, 'LA': 2394, 'OH': 2467, 'MO': 2576, 'IN': 2620, 'PA': 3109, 'MI': 316
1, 'SC': 3936, 'GA': 3963, 'IL': 4350, 'NC': 5091, 'FL': 6185, 'NY': 7318, 'TX': 7396, 'CA': 15388}
```

In [81]:

```
project_data.head(2)
```

Out[81]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Gra
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Gra

2 rows × 21 columns

In [107]:

```
# we use count vectorizer to convert the values into one hot encoded features
```

```
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_scl_st_dict.keys()), lowercase=False, binary=True)
#print(vectorizer.get_feature_names())
vectorizer.fit(project_data['school_state'].values)
print(vectorizer.get_feature_names())


scl_st_one_hot = vectorizer.transform(project_data['school_state'].values)
print("Shape of matrix after one hot encodig ",scl_st_one_hot.shape)
print(scl_st_one_hot)
```

```
['VT', 'WY', 'ND', 'MT', 'RI', 'SD', 'NE', 'DE', 'AK', 'NH', 'WV', 'ME', 'HI', 'DC', 'NM', 'KS', 'I
A', 'ID', 'AR', 'CO', 'MN', 'OR', 'KY', 'MS', 'NV', 'MD', 'CT', 'TN', 'UT', 'AL', 'WI', 'VA', 'AZ',
'NJ', 'OK', 'WA', 'MA', 'LA', 'OH', 'MO', 'IN', 'PA', 'MI', 'SC', 'GA', 'IL', 'NC', 'FL', 'NY', 'TX
', 'CA']
Shape of matrix after one hot encodig  (109248, 51)
  (0, 40) 1
  (1, 47) 1
  (2, 32) 1
  (3, 22) 1
  (4, 49) 1
  (5, 47) 1
  (6, 26) 1
  (7, 44) 1
  (8, 43) 1
  (9, 46) 1
  (10, 50) 1
  (11, 50) 1
  (12, 48) 1
  (13, 34) 1
  (14, 36) 1
  (15, 49) 1
  (16, 47) 1
  (17, 24) 1
  (18, 44) 1
  (19, 38) 1
  (20, 41) 1
  (21, 46) 1
  (22, 50) 1
  (23, 29) 1
  (24, 47) 1
  : :
  (109223, 44) 1
  (109224, 48) 1
  (109225, 46) 1
  (109226, 50) 1
  (109227, 48) 1
  (109228, 37) 1
  (109229, 19) 1
  (109230, 48) 1
  (109231, 32) 1
  (109232, 25) 1
  (109233, 32) 1
  (109234, 48) 1
  (109235, 49) 1
  (109236, 38) 1
  (109237, 40) 1
  (109238, 30) 1
  (109239, 20) 1
  (109240, 25) 1
  (109241, 25) 1
  (109242, 43) 1
  (109243, 39) 1
  (109244, 33) 1
  (109245, 33) 1
  (109246, 48) 1
  (109247, 31) 1
```

clean_categories : categorical data (one hot encoding)

In [82]:

```
# we use count vectorizer to convert the values into one hot encoded features
```

```
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=True
)
vectorizer.fit(project_data['clean_categories'].values)
print(vectorizer.get_feature_names())


categories_one_hot = vectorizer.transform(project_data['clean_categories'].values)
print("Shape of matrix after one hot encodig ",categories_one_hot.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds',
'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (109248, 9)
```

clean_subcategories : subcategorical data (one hot encoding)

In [83]:

```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False, binary=
True)
vectorizer.fit(project_data['clean_subcategories'].values)
print(vectorizer.get_feature_names())


sub_categories_one_hot = vectorizer.transform(project_data['clean_subcategories'].values)
print("Shape of matrix after one hot encodig ",sub_categories_one_hot.shape)
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular',
'Civics_Government', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger',
'SocialSciences', 'PerformingArts', 'CharacterEducation', 'TeamSports', 'Other',
'College_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL
', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences',
'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (109248, 30)
```

In [0]:

```
# Please do the similar feature encoding with state, teacher_prefix and project_grade_category als
o
```

teacher_prefix : categorical data (one hot encoding)

In [84]:

```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
from collections import Counter
my_counter_teacher_prefix = Counter()
for word in project_data['teacher_prefix'].values:
# https://stackoverflow.com/questions/52736900/how-to-solve-the-attribute-error-float-object-has-n
o-attribute-split-in-pyth
    my_counter_teacher_prefix.update(str(word).split())

print(my_counter_teacher_prefix)
```

```
Counter({'Mrs.': 57269, 'Ms.': 38955, 'Mr.': 10648, 'Teacher': 2360, 'Dr.': 13, 'nan': 3})
```

In [85]:

```
tea_pfx_dict = dict(my_counter_teacher_prefix)
sorted_tea_pfx_dict = dict(sorted(tea_pfx_dict.items(), key=lambda kv: kv[1]))

print(sorted_tea_pfx_dict)
```

```
{'nan': 3, 'Dr.': 13, 'Teacher': 2360, 'Mr.': 10648, 'Ms.': 38955, 'Mrs.': 57269}
```

In [86]:

```
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_tea_pfx_dict.keys()), lowercase=False, binary=
True)
#print(vectorizer.get_feature_names())
#https://stackoverflow.com/questions/52736900/how-to-solve-the-attribute-error-float-object-has-no
-attribute-split-in-pyth
vectorizer.fit(project_data['teacher_prefix'].astype(str).values)
print(vectorizer.get_feature_names())

# https://stackoverflow.com/questions/52736900/how-to-solve-the-attribute-error-float-object-has-n
o-attribute-split-in-pyth
tea_pfx_one_hot = vectorizer.transform(project_data['teacher_prefix'].astype(str).values)
print("Shape of matrix after one hot encodig ",tea_pfx_one_hot.shape)
print(tea_pfx_one_hot)
```

```
['nan', 'Dr.', 'Teacher', 'Mr.', 'Ms.', 'Mrs.']
Shape of matrix after one hot encodig  (109248, 6)
  (27, 2) 1
  (75, 2) 1
  (82, 2) 1
  (88, 2) 1
  (201, 2) 1
  (217, 2) 1
  (227, 2) 1
  (285, 2) 1
  (367, 2) 1
  (410, 2) 1
  (412, 2) 1
  (432, 2) 1
  (495, 2) 1
  (549, 2) 1
  (610, 2) 1
  (641, 2) 1
  (689, 2) 1
  (784, 2) 1
  (823, 2) 1
  (839, 2) 1
  (874, 2) 1
  (929, 2) 1
  (931, 2) 1
  (953, 2) 1
  (996, 2) 1
  : :
  (108068, 2) 1
  (108071, 2) 1
  (108095, 2) 1
  (108142, 2) 1
  (108202, 2) 1
  (108344, 2) 1
  (108348, 2) 1
  (108366, 2) 1
  (108402, 2) 1
  (108414, 2) 1
  (108419, 2) 1
  (108616, 2) 1
  (108653, 2) 1
  (108658, 2) 1
  (108743, 2) 1
  (108806, 2) 1
  (108816, 2) 1
  (108881, 2) 1
  (109024, 2) 1
  (109069, 2) 1
  (109129, 2) 1
  (109160, 2) 1
  (109217, 2) 1
  (109219, 2) 1
  (109221, 2) 1
```

1.4.2 Vectorizing Text data

1.4.2.1 Bag of words

In [87]:

```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer = CountVectorizer(min_df=10)
text_bow = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encodig ",text_bow.shape)
```

```
Shape of matrix after one hot encodig  (109248, 16623)
```

1.4.2.2 Bag of Words on `project_title`

In [94]:

```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
# done above
```

In [88]:

```
# Similarly you can vectorize for title also
# We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer = CountVectorizer(min_df=10)
text_til_bow = vectorizer.fit_transform(preprocessed_title)
print("Shape of matrix after one hot encodig ",text_til_bow.shape)
```

```
Shape of matrix after one hot encodig  (109248, 3329)
```

1.4.2.3 TFIDF vectorizer

In [89]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
text_tfidf = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encodig ",text_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (109248, 16623)
```

1.4.2.4 TFIDF Vectorizer on `project_title`

In [90]:

```
# Similarly you can vectorize for title also
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
text_tfidf_title = vectorizer.fit_transform(preprocessed_title)
print("Shape of matrix after one hot encodig ",text_tfidf_title.shape)
```

```
Shape of matrix after one hot encodig  (109248, 3329)
```

1.4.2.5 Using Pretrained Models: Avg W2V

In [91]:

```
# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039
def loadGloveModel(gloveFile):
    print ("Loading Glove Model")
    f = open(gloveFile,'r', encoding="utf8")
    model = {}
    for line in tqdm(f):
        splitLine = line.split()
        word = splitLine[0]
```

```
        embedding = np.array([float(val) for val in splitLine[1:]])
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
#model = loadGloveModel('glove.42B.300d.txt')
model =
loadGloveModel('D:\\Studies\\Hadoop\\Python\\AppliedAICourse\\Notes\\17_REAL_PROBLEM_PREDICT_RATIN(
_AMAZON\\ASSIGNMENT-2 Apply t-SNE\\Assignments_DonorsChoose\\Assignments_DC_2018\\glove.42B.300d\\
glove.42B.300d.txt')

'''
# ============================
Output:

Loading Glove Model
1917495it [06:32, 4879.69it/s]
Done. 1917495  words loaded!

# ============================
'''
'''
words = []
#for i in preproced_texts:
for i in preprocessed_essays:
    words.extend(i.split(' '))

#for i in preproced_titles:
for i in preprocessed_title:
    words.extend(i.split(' '))
print("all the words in the coupus", len(words))
words = set(words)
print("the unique words in the coupus", len(words))

inter_words = set(model.keys()).intersection(words)
print("The number of words that are present in both glove vectors and our coupus", \
      len(inter_words),"(",np.round(len(inter_words)/len(words)*100,3),"%)")

words_courpus = {}
words_glove = set(model.keys())
for i in words:
    if i in words_glove:
        words_courpus[i] = model[i]
print("word 2 vec length", len(words_courpus))


# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-sa
ve-and-load-variables-in-python/

import pickle
with open('glove_vectors', 'wb') as f:
    pickle.dump(words_courpus, f)

'''
```

```
Loading Glove Model
```

```
1917495it [04:20, 7366.13it/s]
```

```
Done. 1917495  words loaded!
```

Out[91]:

```
'\nwords = []\n#for i in preproced_texts:\nfor i in preprocessed_essays:\n
words.extend(i.split(\' \'))\n\n#for i in preproced_titles:\nfor i in preprocessed_title:\n
words.extend(i.split(\' \'))\nprint("all the words in the coupus", len(words))\nwords =
set(words)\nprint("the unique words in the coupus", len(words))\n\ninter_words =
set(model.keys()).intersection(words)\nprint("The number of words that are present in both glove v
ectors and our coupus",       len(inter_words),"
(",np.round(len(inter_words)/len(words)*100,3),"%)")\n\nwords_courpus = {}\nwords_glove =
set(model.keys())\nfor i in words:\n    if i in words_glove:\n        words_courpus[i] = model[i]\r
print("word 2 vec length", len(words_courpus))\n\n\n# stronging variables into pickle files python
: http://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-python/\n\nimport pic
kle\nwith open(\'glove_vectors\', \'wb\') as f:\n    pickle.dump(words_courpus, f)\n\n'
```

In [85]:

```
words_text = []
#for i in preproced_texts:
for i in preprocessed_essays:
    words_text.extend(i.split(' '))

#for i in preproced_titles:
#for i in preprocessed_title:
#    words.extend(i.split(' '))

print("all the words in the coupus", len(words_text))
words_text = set(words_text)
print("the unique words in the coupus", len(words_text))

inter_words_text = set(model.keys()).intersection(words_text)
print("The number of words that are present in both glove vectors and our coupus", \
      len(inter_words_text),"(",np.round(len(inter_words_text)/len(words_text)*100,3),"%)")

words_courpus_text = {}
words_glove_text = set(model.keys())
for i in words_text:
    if i in words_glove_text:
        words_courpus_text[i] = model[i]
print("word 2 vec length", len(words_courpus_text))


# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-sa
ve-and-load-variables-in-python/

import pickle
with open('glove_vectors', 'wb') as f:
    pickle.dump(words_courpus_text, f)
```

```
all the words in the coupus 16540843
the unique words in the coupus 56381
The number of words that are present in both glove vectors and our coupus 49637 ( 88.039 %)
word 2 vec length 49637
```

In [92]:

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-sa
ve-and-load-variables-in-python/
# make sure you have the glove_vectors file
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words_text =  set(model.keys())
```

In [93]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words_text:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors.append(vector)

print(len(avg_w2v_vectors))
print(len(avg_w2v_vectors[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 109248/109248
[00:35<00:00, 3079.48it/s]

109248
300
```

1.4.2.6 Using Pretrained Models: AVG W2V on `project_title`

In [94]:

```
# Similarly you can vectorize for title also
words_title = []

for i in preprocessed_title:
    words_title.extend(i.split(' '))

print("all the words in the coupus", len(words_title))
words_title = set(words_title)
print("the unique words in the coupus", len(words_title))

inter_words_title = set(model.keys()).intersection(words_title)
print("The number of words that are present in both glove vectors and our coupus", \
      len(inter_words_title),"(",np.round(len(inter_words_title)/len(words_title)*100,3),"%)")

words_courpus_title = {}
words_glove_title = set(model.keys())
for i in words_title:
    if i in words_glove_title:
        words_courpus_title[i] = model[i]
print("word 2 vec length", len(words_courpus_title))


# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-sa
ve-and-load-variables-in-python/

import pickle
with open('glove_vectors', 'wb') as f:
    pickle.dump(words_courpus_title, f)
```

```
all the words in the coupus 473570
the unique words in the coupus 16903
The number of words that are present in both glove vectors and our coupus 14051 ( 83.127 %)
word 2 vec length 14051
```

In [95]:

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-sa
ve-and-load-variables-in-python/
# make sure you have the glove_vectors file
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words_title =  set(model.keys())
```

In [96]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_title = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_title): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words_title:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_title.append(vector)

print(len(avg_w2v_vectors_title))
print(len(avg_w2v_vectors_title[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 109248/109248
[00:02<00:00, 51718.43it/s]

109248
300
```

1.4.2.7 Using Pretrained Models: TFIDF weighted W2V

In [97]:

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-sa
ve-and-load-variables-in-python/
# make sure you have the glove_vectors file
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words_text =  set(model.keys())
```

In [98]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_essays)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [99]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words_text) and (word in tfidf_words):
            #vec = model[word] # getting the vector for each word
            #print(type(model))
            vec = model[word] # getting the vector for each word

            # here we are multiplying idf value(dictionary[word]) and
            # the tf value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors.append(vector)

print(len(tfidf_w2v_vectors))
print(len(tfidf_w2v_vectors[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 109248/109248
[04:05<00:00, 445.25it/s]
```

```
109248
300
```

1.4.2.9 Using Pretrained Models: TFIDF weighted W2V on `project_title`

In [0]:

```
# Similarly you can vectorize for title also
```

In [100]:

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-sa
ve-and-load-variables-in-python/
# make sure you have the glove_vectors file
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words_title =  set(model.keys())
```

In [101]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_title)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [102]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_title = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_title): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words_title) and (word in tfidf_words):
            #vec = model[word] # getting the vector for each word
            #print(type(model))
            vec = model[word] # getting the vector for each word

            # here we are multiplying idf value(dictionary[word]) and
            # the tf value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_title.append(vector)

print(len(tfidf_w2v_vectors_title))
print(len(tfidf_w2v_vectors_title[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 109248/109248
[00:04<00:00, 26062.19it/s]
```

```
109248
300
```

1.4.3 Vectorizing Numerical features

In [103]:

```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-
learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ... 399.   287.
73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

price_scalar = StandardScaler()
price_scalar.fit(project_data['price'].values.reshape(-1,1)) # finding the mean and standard
deviation of this data
print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0])}")

# Now standardize the data with above maen and variance.
price_standardized = price_scalar.transform(project_data['price'].values.reshape(-1, 1))
```

```
Mean : 298.1193425966608, Standard deviation : 367.49634838483496
```

In [104]:

```
price_standardized
```

Out[104]:

```
array([[-0.3905327 ],
       [ 0.00239637],
       [ 0.59519138],
       ...,
       [-0.15825829],
       [-0.61243967],
       [-0.51216657]])
```

1.4.4 Merging all the above features

- we need to merge all the numerical vectors i.e catogorical, text, numerical vectors

In [105]:

```
print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(text_bow.shape)
print(price_standardized.shape)
```

```
(109248, 9)
(109248, 30)
(109248, 16623)
(109248, 1)
```

In [108]:

```
print(scl_st_one_hot.shape) # school_state
print(tea_pfx_one_hot.shape) # teacher_prefix
print(text_til_bow.shape) # project title bow
print(text_tfidf.shape) # project_essay tfidf
print(text_tfidf_title.shape) # project_title tfidf
```

```
(109248, 51)
(109248, 6)
(109248, 3329)
(109248, 16623)
(109248, 3329)
```

In [109]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X = hstack((categories_one_hot, sub_categories_one_hot, text_bow, price_standardized))
#X = hstack((categories_one_hot, sub_categories_one_hot, text_bow,
price_standardized,scl_st_one_hot,tea_pfx_one_hot,text_til_bow,text_tfidf,text_tfidf_title))
X.shape
```

Out[109]:

```
(109248, 16663)
```

In [110]:

```
print(type(X))
```

```
<class 'scipy.sparse.coo.coo_matrix'>
```

In [121]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
#X = hstack((categories_one_hot, sub_categories_one_hot, text_bow, price_standardized))
X_CAT_NUM_TITLE_BOW = hstack(( scl_st_one_hot, categories_one_hot, sub_categories_one_hot, tea_pfx_
one_hot, price_standardized, text_til_bow))
```

```
print(X_CAT_NUM_TITLE_BOW.shape)
print(type(X_CAT_NUM_TITLE_BOW))

#X_CAT_NUM_TITLE_BOW_5000=X_CAT_NUM_TITLE_BOW[:5000]

# resize coo matrix |
https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.coo_matrix.resize.html#scipy.spa
oo_matrix.resize
X_CAT_NUM_TITLE_BOW.resize(5000,3426)

print(X_CAT_NUM_TITLE_BOW.shape)
print(type(X_CAT_NUM_TITLE_BOW))
```

```
(109248, 3426)
<class 'scipy.sparse.coo.coo_matrix'>
(5000, 3426)
<class 'scipy.sparse.coo.coo_matrix'>
```

Assignment 2: Apply TSNE

If you are using any code snippet from the internet, you have to provide the reference/citations, as we did in the above cells. Otherwise, it will be treated as plagiarism without citations.

1. In the above cells we have plotted and analyzed many features. Please observe the plots and write the observations in markdown cells below every plot.
2. EDA: Please complete the analysis of the feature: teacher_number_of_previously_posted_projects
3. Build the data matrix using these features
   - school_state : categorical data (one hot encoding)
   - clean_categories : categorical data (one hot encoding)
   - clean_subcategories : categorical data (one hot encoding)
   - teacher_prefix : categorical data (one hot encoding)
   - project_title : text data (BOW, TFIDF, AVG W2V, TFIDF W2V)
   - price : numerical
   - teacher_number_of_previously_posted_projects : numerical
4. Now, plot FOUR t-SNE plots with each of these feature sets.
   A. categorical, numerical features + project_title(BOW)
   B. categorical, numerical features + project_title(TFIDF)
   C. categorical, numerical features + project_title(AVG W2V)
   D. categorical, numerical features + project_title(TFIDF W2V)
5. Concatenate all the features and Apply TNSE on the final data matrix
6. Note 1: The TSNE accepts only dense matrices
7. Note 2: Consider only 5k to 6k data points to avoid memory issues. If you run into memory error issues, reduce the number of data points but clearly state the number of datat-poins you are using

2.1 TSNE with `BOW` encoding of `project_title` feature

steps : aggregate all the features, convert it into dense matrix, standardize the matrix, apply tSNE

In [0]:

```
# please write all of the code with proper documentation and proper titles for each subsection
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

In [119]:

```
#project_data_title_5k = text_til_bow[:5000]
## print(project_data_title_5k)
#print(project_data_title_5k.shape)
#print(type(project_data_title_5k))
#
##convert to dense matrix
```

```
#project_data_title_dense_5k = project_data_title_5k.todense() # converrted sparse matrix to dense
matrix(type matrix)
#project_data_title_array_5k = project_data_title_5k.toarray()
#print("dense")
#print(project_data_title_dense_5k)
#print(project_data_title_dense_5k.shape)
#print(type(project_data_title_dense_5k))
#
#print("array")
#print(project_data_title_array_5k)
#print(project_data_title_array_5k.shape)
#print(type(project_data_title_array_5k))
#
## difference between todense and toarray
## https://stackoverflow.com/questions/30416695/numpy-and-scipy-difference-between-todense-and-toa
rray
## toarray returns an ndarray; todense returns a matrix. If you want a matrix, use todense; otherw
ise, use toarray.
```

```
(5000, 3329)
<class 'scipy.sparse.csr.csr_matrix'>
dense
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
(5000, 3329)
<class 'numpy.matrixlib.defmatrix.matrix'>
array
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
(5000, 3329)
<class 'numpy.ndarray'>
```

In [196]:

```
print(type(X_CAT_NUM_TITLE_BOW))
print(X_CAT_NUM_TITLE_BOW.shape)

X_CAT_NUM_TITLE_BOW_array = X_CAT_NUM_TITLE_BOW.toarray()
print(type(X_CAT_NUM_TITLE_BOW_array))
print(X_CAT_NUM_TITLE_BOW_array.shape)

y_project_approve = project_data['project_is_approved']

y_project_approve_5K =y_project_approve[:5000]
```

```
<class 'scipy.sparse.coo.coo_matrix'>
(5000, 3426)
<class 'numpy.ndarray'>
(5000, 3426)
```

In [197]:

```
y_project_approve_5K.shape
```

Out[197]:

```
(5000,)
```

In [198]:

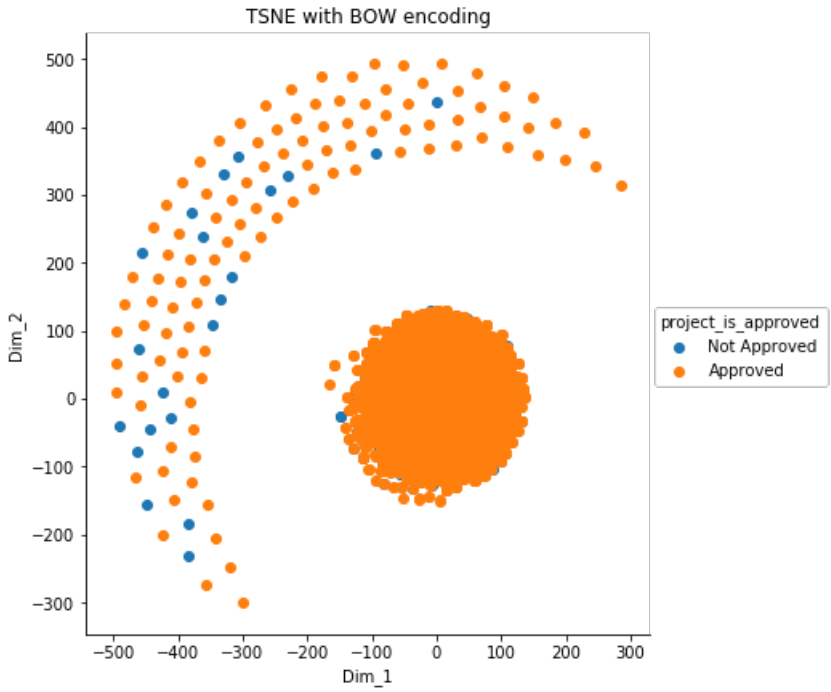
```
## Data-preprocessing: Standardizing the data
#
from sklearn.preprocessing import StandardScaler
```

```
standardized_data_BOW = StandardScaler().fit_transform(X_CAT_NUM_TITLE_BOW_array) # standardize
the dense matrix
print(standardized_data_BOW.shape)
```

```
(5000, 3426)
```

In [199]:

```
from sklearn.manifold import TSNE
import seaborn as sn
# Picking the top 5000 points as TSNE takes a lot of time for 15K points

model = TSNE(n_components=2,perplexity=30, random_state=0,learning_rate=200,n_iter=10000)
# configuring the parameteres
# the number of components = 2
# default perplexity = 30
# default learning rate = 200
# default Maximum number of iterations for the optimization = 1000

tsne_data = model.fit_transform(standardized_data_BOW)

# creating a new data frame which help us in ploting the result data
tsne_data = np.vstack((tsne_data.T, y_project_approve_5K)).T
tsne_df = pd.DataFrame(data=tsne_data, columns=("Dim_1", "Dim_2", "project_is_approved"))

# Ploting the result of tsne
g=sn.FacetGrid(tsne_df, hue="project_is_approved", size=6).map(plt.scatter, 'Dim_1', 'Dim_2').add_l
egend()

legend = g._legend
legend.set_title("project_is_approved")
for t, l in zip(legend.texts,("Not Approved", "Approved")):
    t.set_text(l)

plt.title("TSNE with BOW encoding")
plt.show()
```



Summary

TSNE with BOW encoding for project title, for,perplexity=30, random_state=0,learning_rate=200,n_iter=10000, we could see, there is definite distinguish between Approved and non approved project.

2.2 TSNE with `TFIDF` encoding of `project_title` feature

In [0]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

In [201]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
#X = hstack((categories_one_hot, sub_categories_one_hot, text_bow, price_standardized))
X_CAT_NUM_TITLE_TFIDF = hstack(( scl_st_one_hot, categories_one_hot, sub_categories_one_hot, tea_pf
x_one_hot, price_standardized, text_tfidf_title))
print(X_CAT_NUM_TITLE_TFIDF.shape)
print(type(X_CAT_NUM_TITLE_TFIDF))


# resize coo matrix |
https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.coo_matrix.resize.html#scipy.spa
oo_matrix.resize
X_CAT_NUM_TITLE_TFIDF.resize(5000,3426)

print(X_CAT_NUM_TITLE_TFIDF.shape)
print(type(X_CAT_NUM_TITLE_TFIDF))
```

```
(109248, 3426)
<class 'scipy.sparse.coo.coo_matrix'>
(5000, 3426)
<class 'scipy.sparse.coo.coo_matrix'>
```

In [203]:

```
print(type(X_CAT_NUM_TITLE_TFIDF))
print(X_CAT_NUM_TITLE_TFIDF.shape)

X_CAT_NUM_TITLE_TFIDF_array = X_CAT_NUM_TITLE_TFIDF.toarray()
print(type(X_CAT_NUM_TITLE_TFIDF_array))
print(X_CAT_NUM_TITLE_TFIDF_array.shape)

y_project_approve = project_data['project_is_approved']

y_project_approve_5K =y_project_approve[:5000]
```

```
<class 'scipy.sparse.coo.coo_matrix'>
(5000, 3426)
<class 'numpy.ndarray'>
(5000, 3426)
```

In [204]:

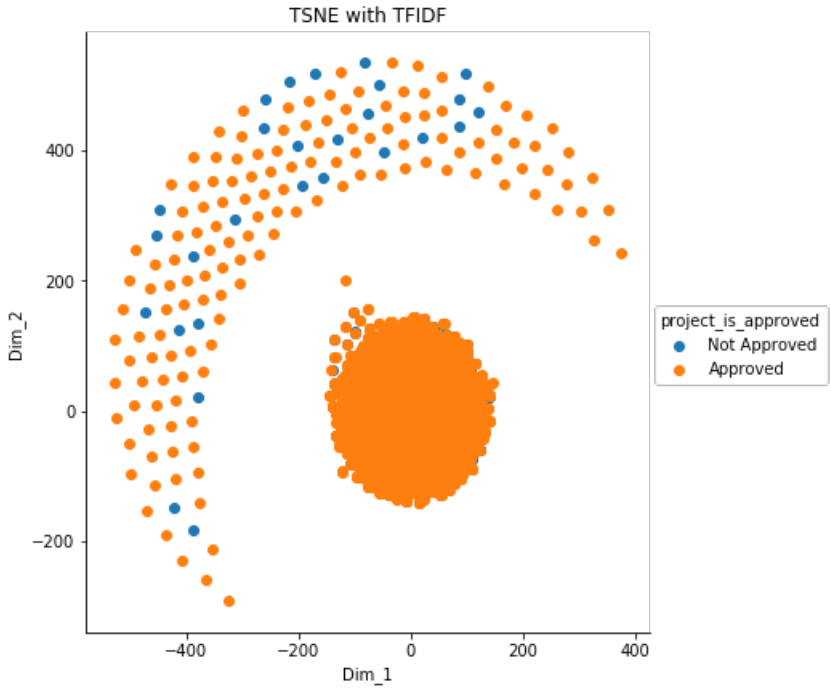
```
## Data-preprocessing: Standardizing the data
#
from sklearn.preprocessing import StandardScaler
standardized_data_TFIDF = StandardScaler().fit_transform(X_CAT_NUM_TITLE_TFIDF_array) #
standardize the dense matrix
print(standardized_data_TFIDF.shape)
```

```
(5000, 3426)
```

In [205]:

```
from sklearn.manifold import TSNE
import seaborn as sn
# Picking the top 5000 points as TSNE takes a lot of time for 15K points

model = TSNE(n_components=2,perplexity=30, random_state=0,learning_rate=200,n_iter=10000)
```

```
# configuring the parameteres
# the number of components = 2
# default perplexity = 30
# default learning rate = 200
# default Maximum number of iterations for the optimization = 1000

tsne_data_TFIDF = model.fit_transform(standardized_data_TFIDF)

# creating a new data frame which help us in ploting the result data
tsne_data_TFIDF = np.vstack((tsne_data_TFIDF.T, y_project_approve_5K)).T
tsne_df_TFIDF = pd.DataFrame(data=tsne_data_TFIDF, columns=("Dim_1", "Dim_2",
"project_is_approved"))

# Ploting the result of tsne
g=sn.FacetGrid(tsne_df_TFIDF, hue="project_is_approved", size=6).map(plt.scatter, 'Dim_1', 'Dim_2')
.add_legend()

legend = g._legend
legend.set_title("project_is_approved")
for t, l in zip(legend.texts,("Not Approved", "Approved")):
    t.set_text(l)

plt.title("TSNE with TFIDF")
plt.show()
```



Summary

TSNE with TFIDF encoding for project title, for,perplexity=30, random_state=0,learning_rate=200,n_iter=10000, we could see, there is definite distinguish between Approved and non approved project.

2.3 TSNE with `AVG W2V` encoding of `project_title` feature

In [0]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

In [207]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
```

```
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
#X = hstack((categories_one_hot, sub_categories_one_hot, text_bow, price_standardized))
X_CAT_NUM_TITLE_AvgW2V = hstack(( scl_st_one_hot, categories_one_hot, sub_categories_one_hot,
tea_pfx_one_hot, price_standardized, avg_w2v_vectors_title))
print(X_CAT_NUM_TITLE_AvgW2V.shape)
print(type(X_CAT_NUM_TITLE_AvgW2V))

# resize coo matrix |
https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.coo_matrix.resize.html#scipy.spa
oo_matrix.resize
X_CAT_NUM_TITLE_AvgW2V.resize(5000,397)

print(X_CAT_NUM_TITLE_AvgW2V.shape)
print(type(X_CAT_NUM_TITLE_AvgW2V))
```

```
(109248, 397)
<class 'scipy.sparse.coo.coo_matrix'>
(5000, 397)
<class 'scipy.sparse.coo.coo_matrix'>
```

In [208]:

```
print(type(X_CAT_NUM_TITLE_AvgW2V))
print(X_CAT_NUM_TITLE_AvgW2V.shape)

X_CAT_NUM_TITLE_AvgW2V_array = X_CAT_NUM_TITLE_AvgW2V.toarray()
print(type(X_CAT_NUM_TITLE_AvgW2V_array))
print(X_CAT_NUM_TITLE_AvgW2V_array.shape)

y_project_approve = project_data['project_is_approved']

y_project_approve_5K =y_project_approve[:5000]
```

```
<class 'scipy.sparse.coo.coo_matrix'>
(5000, 397)
<class 'numpy.ndarray'>
(5000, 397)
```

In [209]:

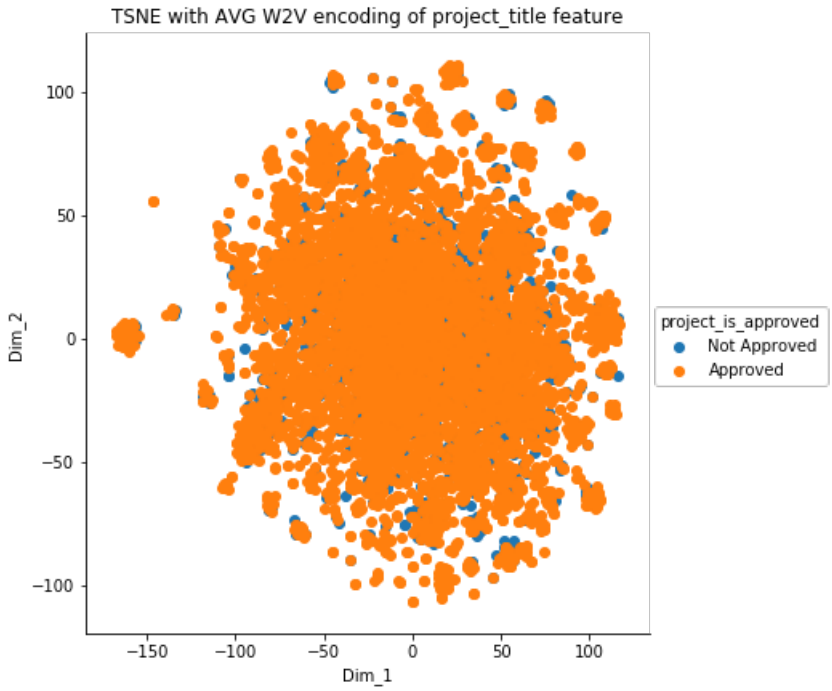
```
## Data-preprocessing: Standardizing the data
#
from sklearn.preprocessing import StandardScaler
standardized_data_AvgW2V = StandardScaler().fit_transform(X_CAT_NUM_TITLE_AvgW2V_array) #
standardize the dense matrix
print(standardized_data_AvgW2V.shape)
```

```
(5000, 397)
```

In [210]:

```
from sklearn.manifold import TSNE
import seaborn as sn
# Picking the top 5000 points as TSNE takes a lot of time for 15K points

model = TSNE(n_components=2,perplexity=30, random_state=0,learning_rate=200,n_iter=10000)
# configuring the parameteres
# the number of components = 2
# default perplexity = 30
# default learning rate = 200
# default Maximum number of iterations for the optimization = 1000

tsne_data_AvgW2V = model.fit_transform(standardized_data_AvgW2V)

# creating a new data frame which help us in ploting the result data
tsne_data_AvgW2V = np.vstack((tsne_data_AvgW2V.T, y_project_approve_5K)).T
tsne_df_AvgW2V = pd.DataFrame(data=tsne_data_AvgW2V, columns=("Dim_1", "Dim_2", "project_is_approve
d"))

# Ploting the result of tsne
g=sn.FacetGrid(tsne_df_AvgW2V, hue="project_is_approved", size=6).map(plt.scatter, 'Dim_1', 'Dim_2'
```

```
).add_legend()
legend = g._legend
legend.set_title("project_is_approved")
for t, l in zip(legend.texts,("Not Approved", "Approved")):
    t.set_text(l)

plt.title("TSNE with AVG W2V encoding of project_title feature")
plt.show()
```



Summary

TSNE with Average Word to Vector encoding for project title, for,perplexity=30, random_state=0,learning_rate=200,n_iter=10000, we could see, there is no distinguish between Approved and non approved project.

2.4 TSNE with `TFIDF Weighted W2V` encoding of `project_title` feature

In [0]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

In [211]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
#X = hstack((categories_one_hot, sub_categories_one_hot, text_bow, price_standardized))
X_CAT_NUM_TITLE_WeigW2V = hstack(( scl_st_one_hot, categories_one_hot, sub_categories_one_hot,
tea_pfx_one_hot, price_standardized, tfidf_w2v_vectors_title))
print(X_CAT_NUM_TITLE_WeigW2V.shape)
print(type(X_CAT_NUM_TITLE_WeigW2V))

# resize coo matrix |
https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.coo_matrix.resize.html#scipy.spa
oo_matrix.resize
X_CAT_NUM_TITLE_WeigW2V.resize(5000,397)

print(X_CAT_NUM_TITLE_WeigW2V.shape)
print(type(X_CAT_NUM_TITLE_WeigW2V))
```

```
(109248, 397)
<class 'scipy.sparse.coo.coo_matrix'>
(5000, 397)
<class 'scipy.sparse.coo.coo_matrix'>
```

In [212]:

```
print(type(X_CAT_NUM_TITLE_WeigW2V))
print(X_CAT_NUM_TITLE_WeigW2V.shape)

X_CAT_NUM_TITLE_WeigW2V_array = X_CAT_NUM_TITLE_WeigW2V.toarray()
print(type(X_CAT_NUM_TITLE_WeigW2V_array))
print(X_CAT_NUM_TITLE_WeigW2V_array.shape)

y_project_approve = project_data['project_is_approved']

y_project_approve_5K =y_project_approve[:5000]
```

```
<class 'scipy.sparse.coo.coo_matrix'>
(5000, 397)
<class 'numpy.ndarray'>
(5000, 397)
```

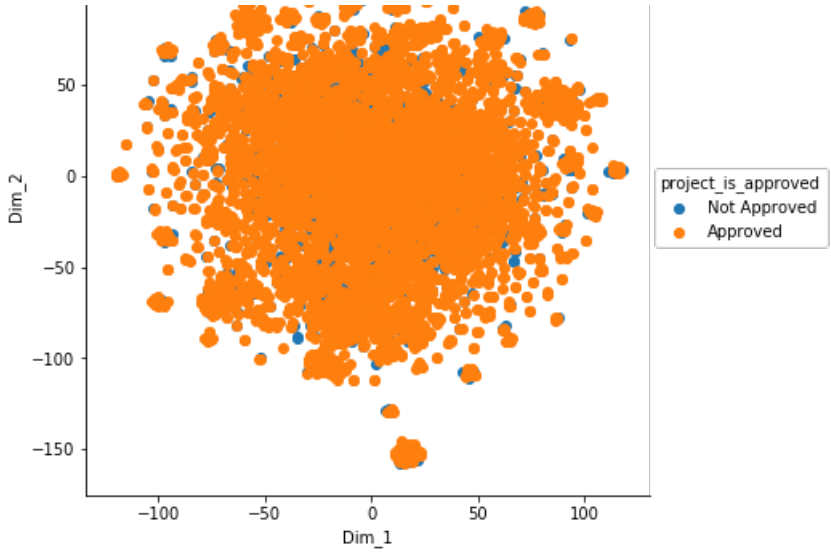
In [213]:

```
## Data-preprocessing: Standardizing the data
#
from sklearn.preprocessing import StandardScaler
standardized_data_WeigW2V = StandardScaler().fit_transform(X_CAT_NUM_TITLE_WeigW2V_array) #
standardize the dense matrix
print(standardized_data_WeigW2V.shape)
```

```
(5000, 397)
```

In [215]:

```
from sklearn.manifold import TSNE
import seaborn as sn
# Picking the top 5000 points as TSNE takes a lot of time for 15K points

model = TSNE(n_components=2,perplexity=30, random_state=0,learning_rate=200,n_iter=10000)
# configuring the parameteres
# the number of components = 2
# default perplexity = 30
# default learning rate = 200
# default Maximum number of iterations for the optimization = 1000

tsne_data_WeigW2V = model.fit_transform(standardized_data_WeigW2V)

# creating a new data frame which help us in ploting the result data
tsne_data_WeigW2V = np.vstack((tsne_data_WeigW2V.T, y_project_approve_5K)).T
tsne_df_WeigW2V = pd.DataFrame(data=tsne_data_WeigW2V, columns=("Dim_1", "Dim_2",
"project_is_approved"))

# Ploting the result of tsne
g = sn.FacetGrid(tsne_df_WeigW2V, hue="project_is_approved", size=6).map(plt.scatter, 'Dim_1', 'Dim
_2').add_legend()

legend = g._legend
legend.set_title("project_is_approved")
for t, l in zip(legend.texts,("Not Approved", "Approved")):
    t.set_text(l)

plt.title("TSNE with TFIDF Weighted W2V")
plt.show()
```



Summary

TSNE with TFIDF Weighed W2V encoding for project title, for,perplexity=30, random_state=0,learning_rate=200,n_iter=10000, we could see, there is no distinguish between Approved and non approved project.

Concatenate all the features and Apply TNSE on the final data matrix

In [216]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
#X = hstack((categories_one_hot, sub_categories_one_hot, text_bow, price_standardized))
X_CAT_NUM_TITLE_All = hstack(( scl_st_one_hot, categories_one_hot, sub_categories_one_hot, tea_pfx_
one_hot, price_standardized, text_til_bow,text_tfidf_title,avg_w2v_vectors_title,
tfidf_w2v_vectors_title))
print(X_CAT_NUM_TITLE_All.shape)
print(type(X_CAT_NUM_TITLE_All))

# resize coo matrix |
https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.coo_matrix.resize.html#scipy.spa
oo_matrix.resize
X_CAT_NUM_TITLE_All.resize(5000,7355)

print(X_CAT_NUM_TITLE_All.shape)
print(type(X_CAT_NUM_TITLE_All))
```

```
(109248, 7355)
<class 'scipy.sparse.coo.coo_matrix'>
(5000, 7355)
<class 'scipy.sparse.coo.coo_matrix'>
```

In [217]:

```
print(type(X_CAT_NUM_TITLE_All))
print(X_CAT_NUM_TITLE_All.shape)

X_CAT_NUM_TITLE_All_array = X_CAT_NUM_TITLE_All.toarray()
print(type(X_CAT_NUM_TITLE_All_array))
print(X_CAT_NUM_TITLE_All_array.shape)

y_project_approve = project_data['project_is_approved']

y_project_approve_5K =y_project_approve[:5000]
```

```
<class 'scipy.sparse.coo.coo_matrix'>
(5000, 7355)
<class 'numpy.ndarray'>
(5000, 7355)
```

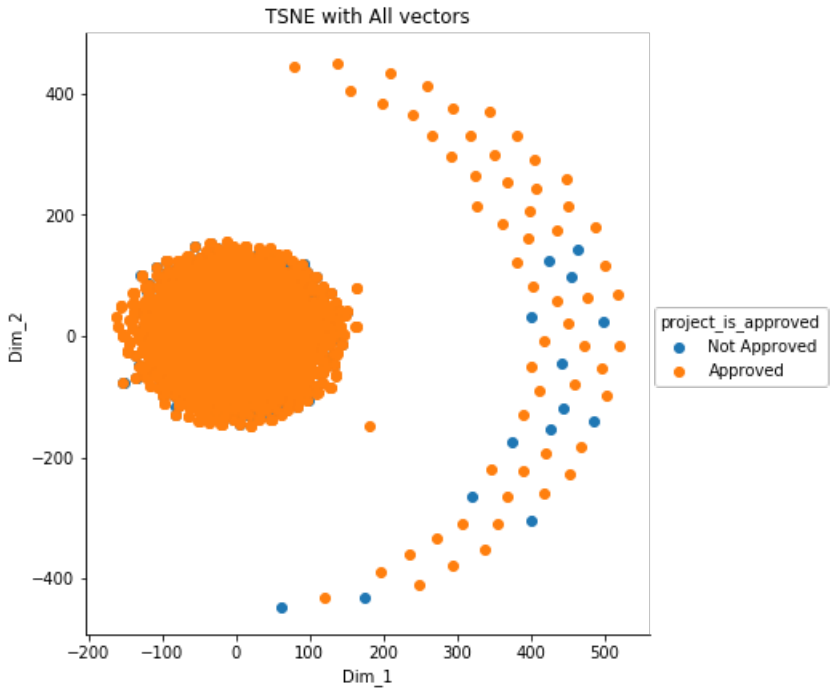
In [218]:

```
## Data-preprocessing: Standardizing the data
#
from sklearn.preprocessing import StandardScaler
standardized_data_All = StandardScaler().fit_transform(X_CAT_NUM_TITLE_All_array) # standardize
the dense matrix
print(standardized_data_All.shape)
```

```
(5000, 7355)
```

In [219]:

```
from sklearn.manifold import TSNE
import seaborn as sn
# Picking the top 5000 points as TSNE takes a lot of time for 15K points

model = TSNE(n_components=2,perplexity=30, random_state=0,learning_rate=200,n_iter=10000)
# configuring the parameteres
# the number of components = 2
# default perplexity = 30
# default learning rate = 200
# default Maximum number of iterations for the optimization = 1000

tsne_data_All = model.fit_transform(standardized_data_All)

# creating a new data frame which help us in ploting the result data
tsne_data_All = np.vstack((tsne_data_All.T, y_project_approve_5K)).T
tsne_df_All = pd.DataFrame(data=tsne_data_All, columns=("Dim_1", "Dim_2", "project_is_approved"))

# Ploting the result of tsne
g=sn.FacetGrid(tsne_df_All, hue="project_is_approved", size=6).map(plt.scatter, 'Dim_1', 'Dim_2').a
dd_legend()

legend = g._legend
legend.set_title("project_is_approved")
for t, l in zip(legend.texts,("Not Approved", "Approved")):
    t.set_text(l)

plt.title("TSNE with All vectors")
plt.show()
```



Summary

TSNE with All the category vector encoding for project title, for perplexity=30, random_state=0 learning_rate=200 n_iter=10000, we

TSNE with All the category vector encoding for project title, for,perplexity=30, random_state=0,learning_rate=200,n_iter=10000, We could see, there is definite distinguish between Approved and non approved project.

2.5 Summary

- Total number of project accepted is quite a good 80%.
- Maximum number of proposal are done by elderly ladies (Mrs)
- Lower Grades gets more proposal, which goes down with increase number of grade(classes).
- Litracy_language and Maths_science are top project, for which Project proposal comes.
- Fewer number of words in Project, more the chances of its approval (except 1)
- no of words in project essay, in range of 250 - 500, has more chances of getting approved.
- For TSNE plot for Project Title, with 5000 data points and below parameter, perplexity = 30; interation = 10k, below encoding give conclusive results, 1) BOW encoding 2) TFIDF encoding and 3) TSNE for all the vector combine.